A Framework for Decentralised Trust Reasoning

Alfarez Abdul-Rahman

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
of the
University of London.

Department of Computer Science
University College London

August 5, 2005

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Abstract

Recent developments in the pervasiveness and mobility of computer systems in open computer networks have invalidated traditional assumptions about trust in computer communications security. In a fundamentally decentralised and open network such as the Internet, the responsibility for answering the question of whether one can trust another entity on the network now lies with the individual agent, and not a priori a decision to be governed by a central authority.

Online agents represent users' digital identities. Thus, we believe that it is reasonable to explore social models of trust for secure agent communication. The thesis of this work is that it is feasible to design and formalise a dynamic model of trust for secure communications based on the properties of social trust.

In showing this, we divide this work into two phases. The aim of the first is to understand the properties and dynamics of social trust and its role in computer systems. To this end, a thorough review of trust, and its supporting concept, reputation, in the social sciences was carried out. We followed this by a rigorous analysis of current trust models, comparing their properties with those of social trust. We found that current models were designed in an ad-hoc basis, with regards to trust properties.

The aim of the second phase is to build a framework for trust reasoning in distributed systems. Knowledge from the previous phase is used to design and formally specify, in Z, a computational trust model. A simple model for the communication of recommendations, the recommendation protocol, is also outlined to complement the model. Finally an analysis of possible threats to the model is carried out.

Elements of this work have been incorporated into Sun's JXTA framework and Ericsson Research's prototype trust model.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

You've got to work on important problems.

Richard W. Hamming

This work is a thought experiment in linking social science issues on trust to the technical design of trust and reputation systems for distributed systems. To guide us we have formulated for ourselves the goal of creating a framework to facilitate trust computation and the exchange of reputational information. The framework we propose, called Ntropi[1], is fully decentralised and based on findings in the social sciences about how people trust one another.

1.1 Why Trust?

The motivation for this work is the security and privacy of users' networks with the following general properties:

Global internetworking The Internet is an example of an open global network, where communication occurs across various boundaries; topological, organisational, political and geographical. In addition, participants in the communication may not be known *a priori* and may never meet physically. Are local systems able to grant access to authorised users effectively while preventing access by unauthorised or potentially malicious users? How does one tell the difference between these two groups of users?

Ubiquitous connectivity The availability of mobile and wireless technology means we are now not only able to connect people together but also connect objects together, and objects to people. This has opened doors to applications that allow objects to be controlled remotely via the network and groups of objects to collaborate in an ad-hoc manner for various tasks. Intermittent connectivity and short-lived relationships are characteristic of such systems. Questions arise as to whether reliability of these ad-hoc components can be ascertained. How does the local security policy of mobile objects cope with potentially encountering and interacting with hundreds of other objects in a very short period of time?

Software as agents Increasingly, users are delegating tasks to software. Examples include automated notification of news items, purchasing items online and matching user preferences.

[1] Networked trust opinions.

> While it relieves us from mundane tasks, the potential for agents to carry out riskier transactions is impeded by uncertainty in the security of such a system. Can the agent be sure that the other agents it communicates with are trustworthy enough (for whatever criteria is important to that communication)? Is the platform or system surrounding that communication safe enough? How does the agent decide if it is?

We believe that trust plays a central role in the security of interactions in the systems described above. Detailed discussion of this is presented later in this chapter. In order to specify the requirements for a model of trust in distributed systems it is essential that we understand the mechanism and dynamics of trust. For this, we investigate the trust literature from the social sciences because trust is a social phenomenon and the area of social sciences is one where trust has been extensively researched. The framework presented in this dissertation is one instantiation of the model that is based on our findings.

The Oxford Dictionary of Modern English [Swa92] defines two interpretations of *trust*:

1. *Confidence in or reliance on some quality or attributes of a person or thing, or the truth of a statement.*
2. *Confidence in the ability and intention of a buyer to pay at a future time for goods supplied without present payment.*

Definition one highlights the social role of trust, that is, trust plays a role within interactions with others. For example, we place trust in public transport every day to take us safely to work and back home to our families. We trust that the route home we take after a late night is one with the least risk of us getting mugged or harassed. Furthermore, we hear trust being emphasised as being the foundation of solid inter-personal relationships.

Definition two suggests that credit-based transactions are based on trust. Without a formal contract, we directly trust that the buyer will pay us the agreed amount by the agreed date. Even when a contract is involved, trust is present, although not directly between the buyer and seller. For example, the seller places his trust in the law and expects the courts to enforce contract law in case the buyer defaults on the contract. Trust can also be extended into other aspects of business. For example, we may trust the supermarkets to sell us good quality food or we may trust that our mechanic will do a good job on our car and not 'pull a fast one' on us. Trust is handled differently in different situations and with different aspects of each situation.

Our social and economic lives now include the 'virtual' world too. We can socialise and buy or sell goods on the Internet, and we make trust decisions when deciding which seller to buy from, or which buyer to sell to [eba, Boy02].

Trust is a phenomenon that has inspired many social scientists and economists. This is evident when one carries out a survey of the relevant fields in the social sciences and economic theory. The appeal is understandable because trust is widely viewed as a vital foundation of human communication and interaction, i.e. society [Luh79]. Yet this important notion has eluded human comprehension through centuries of civilisation, until very recently, when the complexities and volatility of modern society

pushed social scientists and psychologists into investigating trust and its functions in 'lubricating'[2] human interaction.

We must now add another dimension to the problem of comprehending the meaning of trust. That is human interaction[3] through computer networks and distributed systems. The Internet has had a pervasive impact on how people interact, evolving from a medium for sending simple messages to one another into a medium for communication, interaction and control. Furthermore, agent technology [WJ95] enables us to delegate tasks to software that works on our behalf, communicating with other agents and people.

In short, we can sum up the future of society plausibly as a network of communicating and interacting humans and artificial 'agents'.

If trust forms the basis of successful, 'off-line', human interaction then it must also form the basis of our on-line interaction. Artificial agents will need to reason about the trustworthiness of other humans or agents, and humans will need to apply naturally their trust reasoning mechanisms into the artificial environment of the computer. Put another way, the computer is merely an extension of human interaction and therefore there must also exist the ability to transfer human trust reasoning into computers.

1.2 Timeliness of Research

We believe this research is a timely body of work in computer science.

Steve Marsh's Ph.D. dissertation titled "Formalising Trust as a Computational Concept", published in 1994, is the first major body of work on bringing a social trust model to computer systems[4]. This was at the threshold of the 'Internet goldrush' that saw a dramatic rise in the number of users and applications that were deployed. Up until the publication of Marsh's dissertation, trust as a research topic had been investigated in isolation. Primarily trust was the subject of formal analyses of network security protocols and public key infrastructures (PKI). Formal tools such as Burrows et al's logic (BAN logic [BAN90]) broke new ground in giving security analysts a tool to understand the implications of messages and channels in their protocols. Then as enterprise-scale networking became pervasive in industry, a challenge arose where one must give access to corporate resources to potentially unknown entities such as employees from another country, field workers as well as business partners. Some kind of delegation of access rights was required, the notion of a 'capability' [DH65] regained interest and cryptographically signed certificates started being used as capabilities instead of just for verifying identities [E+99].

Sometime during the mid-1990s, work on trust as a stand-alone topic in security began to gain momentum. In 1996 the Center for Discrete Mathematics and Theoretical Computer Science (DIMACS) at Rutgers University organised the first workshop on this topic: the DIMACS Workshop on Trust Management in Networks. Included in the proceedings is an early paper from the work presented in this dissertation [ARH96]. Around this time, papers on trust also began to appear in

[2] Term borrowed from [Lor88].

[3] Throughout this dissertation, we will use the term interaction to include verbal communication as well as other types of messages exchanged between two or more entities on the network (for example a message to a remote object representing a method invocation).

[4] A survey carried out by Graham Klyne identified Marsh's work as being "seminal in computation of trust" [Kly04].

the ACM New Security Paradigms Workshops, a forum where authors are encouraged to present works-in-progress on unconventional ideas in security research [ARH97a].

A parallel development was also occurring in the field of collaborative filtering (whose application is known as recommender systems). In it, user preferences for items in a specific area are used to recommend further items based in preference similarity with other users [RV97]. The notion of relevance to our work here is the sharing and aggregation of opinions to create new information. One of the most high profile applications of collaborative filtering can be seen in amazon.com's recommendation engine [LSY03]. Also of note is the work carried out at MIT by Patti Maes and her research teams, notably her early work with Upendra Shardanand on a music recommendation system [SM95]. Maes and her team extended the idea into general reputation models for online marketplaces for autonomous agents. However these solutions are based on *centralised* aggregation of opinions. As far as we know, the work of Rasmusson and Jansson was one of the first to take this idea and apply it in a distributed environment [RJ96]. Their work introduced the idea that sharing of reputation information can inform distributed trust decision making.

Peer-to-peer applications, or P2P, became widely used at the beginning of the millennium, with Napster leading the pack [drs00] (albeit through a centralised server), followed by systems such as Gnutella [gnu] and Freenet [CMH+02]. Peer based, distributed protocols were also used in mobile ad-hoc networks [RT99], another increasingly popular research area. These technologies point to increasing use of systems that connect individual hosts without centralised control. Whereas P2P applications necessitate direct user interaction, there are also peer-based systems that must make automated decisions, such as routers in ad-hoc networks.

More recently a class of Internet services called social software began making the social networks of its users explicit and visible, using them to drive the design of the technology itself. Examples include weblogs, or 'blogs' (e.g. Blogger [blo]), social networking software (e.g., Friendster [fri] and LinkedIn [lin]) and content syndication (e.g. RSS [Win]). These systems motivate grassroots production of resources, discovery of those resources through trusted networks of contacts, and automation of content receipt through regularly updated 'feeds'.

Thus, we believe this work, to create a framework based upon a decentralised trust model, is a timely body of work because:

- There is increased use of decentralised applications that necessitate decision making by both the user and software. These decisions include assessment of trust of other users or network objects.
- The notion of trust and reputation is now a mainstream research topic but inter-disciplinary efforts are still at their very early stages. Most trust models are based on the system modellers' assumptions of how trust functions.
- Trust and reputation are beginning to form part of the everyday tools used by users online, from purchasing to seeking information or other people.
- Concerns for security, anonymity and identity theft continue to grow, particularly in P2P and wireless communications, and knowing with whom to trust personal information is crucial.

1.3 Security and Trust

The following is Garfinkel and Spafford's definition of 'computer security' [GS96]:

> *A computer is secure if you can depend on it and its software to behave as you expect.*

This traditional definition of security describes the general paradigm which underlies early security research and products deployed in current networks. It highlights three interesting aspects of traditional security approaches:

Absolute security The definition suggests that security must be a complete notion, an all-or-nothing concept. A computer is either secure or not. If vulnerabilities exist in a computer system then it is not secure, therefore not dependable. However, dependable or not, useful work is still being carried out on computer systems that, with their increasing complexity, are exhibiting increasing vulnerabilities despite ongoing efforts to 'patch' these vulnerabilities. Figure 1.1 illustrates the trend in reported vulnerabilities. Thus computer systems that are not considered completely dependable are, at the same time, useful to users who need them in their work. One must then start to question the relationship between each vulnerability, the risks it poses and how it affects the applications of the system. There are *degrees* to these risks, and therefore, the security of computer systems, and making the shades of insecurity apparent will help the user make informed judgements on whether a particular system is suitable, and dependable, for the task she is about to undertake.

Trust in systems The definition also suggests that the subject of dependability here is the computer. The model of security here is one where trust is granted by the user to the computer system itself. This hides the fact that the computer is the product of the people who created, programmed, managed and used it and any vulnerability, intentional or not, can only come from these sources. If a local area network contains very sensitive information yet does not have a firewall to detect intrusion attempts then it is the system administrator's job to fix that – it is not the system that is at fault.

Complete knowledge In order to know what to expect from a computer's behaviour, one must have complete knowledge of the possible behaviours of that computer. This is perhaps beyond the capability of most security practitioners, let alone users, as it entails analysis of every line of code of all software that is installed in the system, and those that it connects to. This idea is related to the notion of absolute security above because to know that we have covered all security vulnerabilities, we must be able to comprehend all possible actions a computer system can take. However, if this requirement is relaxed, and we allow for the uncertain and unknown, then one's risks become an important consideration. The amount of risk to take can then be based on levels of uncertainty in what is possible and what is not, with respect to various security vulnerabilities.

The evolution of communication from standalone systems to open global networks has driven research and thinking behind computer security. Early computers in industry were big number crunchers that were treated as "fortress mainframes". Security of these mainframes were largely physical

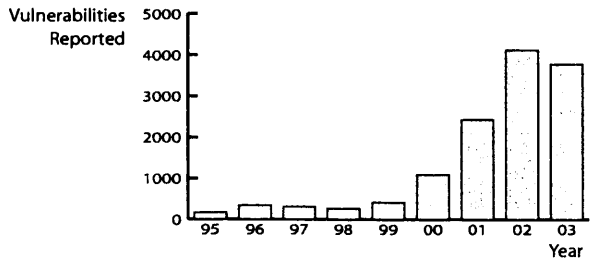


Figure 1.1: Number of reported security vulnerabilities by year, as reported by CERT/CC [Cen04].

where access required the opening of locks on doors in high security buildings. Since access was limited to those given explicit permission, monitoring and control was feasible.

As computer networks became more pervasive and client/server architecture became popular, there was a need to carry out resource access online. This gave rise to online access control mechanisms based on permissions and access control lists like those found in the UNIX operating system. Furthermore, each host on the network can specify which other hosts and users on the network it trusts, allowing these trustees access to the host as though they were actually logged on at the host itself.

These access control mechanisms are based on the assumption that whomever is logged in to the system must be a legitimate user. This is a reasonable assumption for local area networks because, again, control and monitoring is feasible. A network administrator is responsible for registering users on to the network, and the limited number of users meant access logs are not too large to analyse in case of suspected attempted violations of access rights.

However, as networks begin to grow beyond the boundaries of organisational control, the validity of traditional assumptions of its security began to be tested. Large networks of computers are now commonplace and isolated local area networks are now being connected together. Virtual private networks span the globe as organisations spread operations geographically. It is now less certain whether the assumption that a successful login can only be carried out by a legitimate user is a valid one. Take the example of remote login into a virtual private network. Is the user actually the person registered to use the network? Could the password have been stolen by someone? From where is the terminal being used to login from? What is the security policy of the network from which the user is logging in?

Misfeasance can also be a cause of uncertainty. With the multitude of online services now available to the public, and each one of them requiring registration, is it surprising to find that users actually write passwords down on 'post-it' notes and stick them to their screen [ASL97]? Can network administrators trust all their registered users to keep passwords secret?

In addition to large virtual networks, the Internet now presents services providers with potential access to any user on the planet. However, with such an open system comes the uncertainty in the process of authorisation. Verifying an identity becomes a very difficult problem. How can one be certain that a public key certificate belongs to the person or entity stated in the certificate? Can we trust the certifier enough to believe this assertion? Can the same name refer to more than one person? Who are these certification authorities whose certificates are distributed with web browsers by default, and can we trust them? What should we be trusting them for?

A human user may be able to make some reasonable assumptions in relation to the questions above when interacting online, and can make judgements on whether to trust entities on the Internet based on cues provided by information that can be obtained through the medium itself. This task would be much harder for artificial agents to carry out. With the rapid growth of peer-to-peer applications, grid systems and ambient computing, trust related decisions must now be carried out by computer more than ever before. Should a peer host be trusted to cache a portion of a file to make it more available? Can a processor on a grid be trusted to process sensitive data? Is the wireless channel in the next block secure enough for conversations with colleagues about an ongoing project? Will these hosts collude to expose my online behaviour?

The degree of importance such questions have to individual users in different applications will vary. Nevertheless, in an open system where interactions can involve multiple unknown entities, effective policies on trust and security should be made with an understanding of how trust works at the level of the user, i.e. social trust.

The limitations of relying on certainty and objectivity have catalyzed an awareness for the need to better understand trust. We now have better insight into trust; for example we now know that trust is not absolute [BBK94, Ger98] and it is not transitive [CH96].

Trust concerns are also relevant to reasoning about information content [ARH99]. The wide array of information sources now available to the masses will make it difficult to select those that are trustworthy. An example is the rating of content on the World Wide Web, using rating systems such as PICS [KMRT96]. There may be a large number of rating agents and choosing a suitable and trusted rating agent is left up to the end user, or recipient of the content rating information. Here, the user must use his discretion on whether to accept the rating based on whether the rating agent is trusted or not. Evaluating each rating on each document may overwhelm the user so automation for this task is desirable. An example of such a system can be found in [Kha96], where the PolicyMaker [BFL96] trust management software is used to execute the relevant policies according to the appropriate rating agent encountered.

The original security protocols worked within networks with well defined boundaries. Open networks break those boundaries and strain traditional trust assumptions. Thus there is a real need for trust in distributed systems to be better understood.

1.4 Peer to Peer Computing

Peer-to-peer technology, or P2P, refers to systems that work on the assumption that any computer on the network can be both a client and a server of a service, or both a producer and consumer of information.

The Internet works on technology that is inherently peer-based. There are also early applications that, although not termed peer-to-peer, work on the same principle. Usenet is one such example, which has existed since 1979 and is still in use today. However, P2P as a distinct concept for network applications without centralised control has gained momentum in recent years due to increasing availability of distributed systems technology and the popularity of file sharing P2P applications. Furthermore, the ability for any user to encode audio and video content into digital media has also

controversially allowed users to share copyrighted material – users can search for and download music and movies for free, provided someone has gone through the trouble of 'ripping' them from their original sources. P2P applications enable file sharing by opening up the hard disk contents on the machine the application is running on to the outside world through the Internet.

New challenges have now been presented to the designers and users of P2P applications, as follows:

Free riding Free riding happens when a user obtains public resources without payment. Research by Adar and Huberman reported that "almost 70% of Gnutella users share no files, and nearly 50% of all responses are returned by the top 1% of sharing hosts" [AH00]. The effect of this is degraded performance of the system, which is bad for everyone, and exposes the small percentage of contributors (who in effect are acting as centralised servers) to attacks. [5].

Resource quality uncertainty Selection of target or routing peers in most P2P networks is based primarily on performance metrics, such as nearest neighbours or size of bandwidth [AH02]. However, there is uncertainty about the quality of the selected peer and the resources available from it. A well known problem in file sharing networks is the intentional distribution of files whose contents do not match their description or are corrupted. This is a popular strategy by music production companies. The idea is that by saturating popular file sharing networks with bad quality files it will cost downloaders more, in terms of time and effort, to get to good quality copyrighted material. To those who can afford to buy 'legitimate' copies at a store, this would hopefully make purchasing a more attractive option.

Viruses Files may also be maliciously created as trojan horses or viruses, spreading themselves through the P2P network [Van01].

Privacy In some P2P implementations there is a need to hide the contents of communication and/or identities of parties. For example, by storing a controversial document, it may become a target for attack. Thus anonymising systems, such as Freenet [CMH+02], would benefit this application. Such solutions rely on collaborative protocols, such as Chaum's mix-nets [Cha81], and therefore assumes that collaborating nodes are reliable and trustworthy. This assumption is questioned in real implementations as the potential for rogue nodes to be included in the system exists. Thus trust and reputation models to manage this risk have been looked at [DS02]

In networks that view the scenarios above as problems, allowing users to assess trustworthiness in other agents and communicate them as reputational information can reduce the impact of attacks on the network. Nodes that have a reputation for free riding, disseminating bad files, being a source for viruses and leaking private information can be identified. Users then have the choice of not interacting with these nodes.

It can be said that the popularity of P2P applications have spurred the interest in trust research in recent years. Sun Microsystem's P2P platform, JXTA, is an example of a system for which a distributed trust model has been specifically proposed [CY01]. Called Poblano, the model is based

[5] An alternative positive view of free riding has been put forth, arguing that free riding on networks is what makes public P2P networks successful and does not consume resources in the physical sense, e.g. a file doesn't disappear from the source host when it is downloaded [Shi00]

on our work described in this dissertation, published in an earlier paper [ARH97a]. Furthermore issues of trust have also been discussed in the research community (see, for example, [AD01]) and implementations have been considered in various popular P2P platforms such as Gnutella [gnu], Free Haven [Sni00] and Freenet [CMH+02]. Details of their trust models are discussed in Chapter 4.

Related to peer-to-peer technology is the grid network architecture [FK99]. They both have similar goals which are sharing resources spread across multiple individual computers to perform collaborative tasks. However the grid architecture was originally developed in a hierarchical manner with tighter control on security policies and the applications that run on it. The main motivation behind grid projects was giving those who needed it access to high performance computing power that is distributed across multiple supercomputers that are networked together. Researchers have recognised that P2P and the grid share many similarities and the two communities can learn from one another. Thus we are starting to see convergence in the two areas [TT03] and an increase in discussions on trust issues in the grid [DT02].

1.5 Intelligent and Autonomous Agents

Distributed Artificial Intelligence (DAI) is an area concerned with 'agents' that are 'intelligent' and able to function autonomously [WJ95]. These software agents communicate with humans as well as other agents. Agents may also form societies [Dav02], communicating, interacting, transacting and exchanging information like we do in the real world. To function as a society, the same issues affecting human society are involved and this includes the ability to make trust-related judgements about other humans and agents.

There are various risks to agent systems that necessitate some assumption of trustworthiness in the entities involved in interactions. Agents may encounter other agents that may be malicious or unreliable. Encounters with untrustworthy seller agents by a buyer is a classic example [RJ96]. Agents must also be aware of the environment within which they are running. Mobile agents may move from one host to another and there may be uncertainties about the security of each host. For example, is the host secure from tampering? Is the host a machine that is accessible to any user, such as a terminal in a cyber cafe? If the host itself is mobile and communication is wireless, such as a PDA or mobile telephone, then the host itself may be moving through different domains. Can the agent be certain that the current wireless channel is encrypted if sensitive information is being transmitted? Does the geographical location present a high risk of the user being robbed of his mobile terminal, hence the agent itself?

It is important for agents to be able to trust or distrust if they are to survive outside the confines of the laboratory [Mar94a]. Their survival depends on their ability to form opinions about the trustworthiness of other agents, hosts that run the agents and people. Without this ability, agents are being released into the 'real' network with the assumption that all other agents are benevolent and trustworthy. Ultimately, an agent is the product of a human being and owned by a human being. Its actions are in faithful obedience to its owner's orders. Therefore, when we deal with agents, we are dealing with the intentions of human beings, albeit indirectly. Furthermore, it is unreasonable to assume that programmers write absolutely accurate code that guarantees that Byzantine behaviour will not occur. This is why we cannot be certain about the behaviour of agents that we encounter.

According to Peter Neumann, "We might like to believe that we can trust computers more than we can trust people. However, the bottom line is that, because the computer systems are themselves products of human effort, we cannot expect to trust them either – even if we have gone to enormous efforts to increase trustworthiness." [Neu92]. Marsh [Mar94a] stated that an understanding of trust will benefit agents in two ways:

1. By allowing agents to determine whether other agents are trustworthy or not.
2. By allowing agents to assign different trust levels to agents in different situations.

In 1994, Marsh claims that the field of DAI is short-sighted for its lack of explicit consideration for trust [Mar94a]. However, in the short few years that followed, trust research found some friends in DAI and research began to gain momentum. The subject of trust was a major theme in an international workshop[6]. Clearly, there is now an awareness in DAI of the importance of this research area.

1.6 Trust as a Social Phenomenon

Trust makes cooperation possible, and is an essential ingredient in interpersonal and business relationships. It is really something that manifests itself in societies. Trust exists in the space between two interacting agents. If a person lives alone without communication then there will be no need for trust[7] [Luh79]. As soon as we interact, trust comes into play. This is of particular interest because it is a picture that we can also describe distributed systems with – a network of autonomously interacting entities. There is no need for a standalone computer to decide whether to trust or distrust because it (or rather the user) is only involved in interactions with himself. However, if we refine the graininess of our 'standalone' computer concept to include interacting processes in the OS, then trust again comes into play, unless all processes were written by the same programmer. As Thompson said in [Tho84], the only program code we can trust is one which we have written ourselves. It is important to note that Thompson was talking about trust with respect to the program being devoid of intentional malicious code, e.g. trojan horses. Without this qualification of the statement, it is ambiguous.

As soon as the 'standalone' is connected to the network, there is communication with the 'outside world' and thus trust will be called upon to assist it in making decisions pertaining to other parties it interacts with.

This parallel between human societies and distributed computer systems suggests that what we can learn about trust in human societies may be applicable in distributed systems. This is important as the social sciences have tackled many questions about trust. However, although there have been many works by social scientists on trust, there is an evident lack of coherence in their *definition* of trust [Mar94a, MC96]. For example, Bernard Barber [Bar83] characterises trust as being based solely on social structures and not on personal attributes while, Diego Gambetta [Gam88a] considers trust at a more personal level, describing it as a certain type of belief in other persons [MC96].

[6]The workshop on Deception, Fraud and Trust in Agent Societies, Minneapolis/St. Paul, U.S.A., 9/5/1998.

[7]We shall only include man's interaction with his society, and thus exclude his relationship with animals or other inanimate objects.

The reason for the differences is due to how trust was studied – usually as an empirical experiment with a very narrow focus. This is unsurprising as trust is a very difficult concept to grasp. Furthermore, there has been a lack of theoretical analysis of the phenomenon compared to its empirical study [MC96]. As there are many views of trust [Mar94a], each of these narrowly focused experiments enlightens only a small part of the big picture. Sometimes these spotlights of experimental results overlap but, largely, they are isolated. This is clearly unsatisfactory when one wants to understand the concept of trust as a whole. Recognising this problem, McKnight and Chervany made an attempt to put together the pieces by drawing from the literature of trust research in the social sciences and then characterising trust as a set of inter-related constructs [MC96]. Their work encompasses trust findings from both empirical research and theoretical analysis.

Deeper meanings of trust as experienced by humans will be explored further in Chapter 2.

1.7 Trust in Business and Economics

In day to day transactions, trust is displayed on many levels; from the trust of the buyer in the seller (that he is not being sold 'lemons' for example) to the trust of the seller in the currency he receives from the buyer. Although being something central to transactions, trust is also rarely discussed in mainstream economics [Das88]. It is normally assumed that in economic models trust is there when you need it. For example, Dasgupta says that in

> *the standard model of a market economy it is taken for granted that consumers meet their budget constraints: they are not allowed to spend more than their wealth. Moreover, they always deliver the goods and services they said they would. But the model is silent on the rectitude of such agents [Das88, pp. 49].*

In the same paper [Das88], Dasgupta carries out an interesting discussion about trust and its role as a commodity. We will look at the points he raised in the next chapter. One of the points is interesting and worth noting here. According to him, in any given context, the value (or "worthwhileness") of trust can be measured, although there are no obvious units to measure it with. He relates trust to commodities such as knowledge or information in this respect.

Lorenz also gave an interesting viewpoint [Lor88]:

> *if transaction costs are thought of as friction in the economy, then trust can be seen as an extremely effective lubricant.*

Lorenz refers to the ability of parties in transactions or business to reduce costs and overhead when there is trust in one another. This gives them the confidence to do away with contingencies in case the other party behaves opportunistically. Contingencies, usually in the form of contracts, cost money and time. Being able to replace all that intricacy with trust is a very powerful concept.

A clear understanding of trust will allow us to enhance and build solid foundations for commerce. In distributed systems, electronic commerce stands to benefit from this. Trust will also play an important role in another aspect of e-commerce, that is the absence of face-to-face contact and a market that is potentially global in scope.

1.8 Reputation and Recommendation

Reputation is a powerful distributed mechanism of social control. Through word-of-mouth among the members of a community, reputation has the potential to purge society of 'bad' agents and promote 'good' ones. In economic theory, reputation plays an important role. As an example, Tadelis considers reputation itself as something which carries monetary value and can be traded [Tad99]. This reputation is normally in the form of names. Once a company has built up its business its name increases in value. Recognising this, company names can then be bought or sold.

Which agents are bad and which are good is subjective and depends on the collective values of the society and the context within which reputation functions. This is relevant, as we will later examine how reputation information, via word-of-mouth, is used to assess the trustworthiness of agents. In this work a mechanism for propagating and exchanging reputation information is proposed. We call this the Recommendation Protocol. The protocol uses Recommendations that are messages that encapsulate the actual reputation information. These Recommendations can be given out proactively or reactively.

Recommendations have been shown to work rather successfully in distributed system applications. The best known technologies today for doing this are grouped under the name 'collaborative filtering', although the name is slightly misleading as there is little active collaboration involved between entities and there are more recommendations made than filtering. [RV97] contains a good overview of these technologies, which the authors call "recommender systems".

In short, a recommender agent collects recommendations and preferences from other agents about a particular area of interest, music for example, and tries to match these inputs with the profiles of other agents. When there is a match, the recommender agent will provide new recommendations based on this collected profile. So if Alice bought CDs A and B and Bob had bought the same two CDs along with a third, CD C, then the recommender agent may recommend CD C that Bob bought to Alice. The more you contribute to your profile in the recommender agent's database, the better the agent will be at recommending the right choices to you.

This approach, however, is unsuitable for our purpose, which is recommending trust. Generally, recommender systems like the one above are devoid of reputation information. The advantage of this is anonymity of users of the system as personal preferences are hidden from view of other users. However it does not allow users to select other users whose taste they trust more. Furthermore, it is not a distributed solution because it relies on a central recommender agent.

How we tailor this for our own use will be discovered in later sections. It is sufficient here to show that recommendations and reputations are useful tools for this work and that real world implementations of recommendation technology have been built.

1.9 Coping with Uncertainties in Interactions and Information

Although the recommendation and reputation mechanisms are useful tools for managing and reasoning about trust, they are not completely error free. Inherent in any type of distributed system is the inability to obtain complete information and this subjects recommendation and reputation tools to

uncertainties and ambiguities. As we discussed in §1.3, security technology that relies on certainty is ultimately limited in its effectiveness when applied in large open systems because of the uncertain nature of the system. We may wish to build something that delivers complete certainty, to say that an input of x will always produce y, but such a system will only promote false confidence because uncertainty is inherent in all interaction.

What we require is an approach where uncertainty is allowed for. This approach must permit us to function within the grey areas between the endpoints of absolute certainty, dealing with situations that are probable or possible.

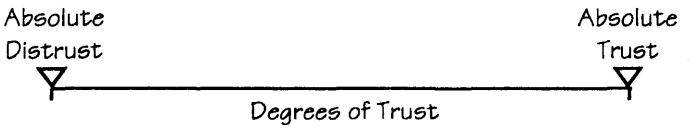


Figure 1.2: Trust is not absolute.

The problem of uncertainty will always exist because there will always be incomplete knowledge. It is beyond our resources and capability to collect all information that is relevant for our actions. Thus we must make judgements based on incomplete information, based on uncertain situations.

In the field of AI and Knowledge Engineering (KE), tools have been developed to handle uncertainty and incompleteness. Examples include probability and possibility theories, fuzzy set logic, rough sets, non-monotonic logics, argumentation and belief functions [KC93].

The concept of uncertainty is relevant to this work in two ways:

- Trust is *used* to cope with uncertainty in distributed system interaction.
- Uncertainty exists in the trust reputation information recommended about other agents.

The complexity of modern society is one factor that can increase uncertainty. As Luhmann points out, for agents that attempt to align themselves with the complexity of the world, their only problem is "the relation of the world as a whole to individual identities within it, and this problem expresses itself as that of the increase in complexity in space and time, manifested as the unimaginable superabundance of its realities and its possibilities" [Luh79]. To cope with this problem, Luhmann posits that trust is called on to reduce the complexity of society. This underlines the first aspect of trust that is relevant to this work above.

The second point is related to when an agent receives recommendations from other agents. Lack of knowledge places uncertainty in the received information. Can the recommender be relied upon for this information? What are the motives of the recommender for making this recommendation? Is he using the same standard of judgement and, if not, how do I adjust his recommendation accordingly to suit my standards?

1.10 Decentralisation

The structure of various network topologies can be generalised into either hierarchical or peer-based. Examples of hierarchical systems include the Internet naming system [Moc87] and the X.509 Di-

rectory Authentication Framework [ITU97]. The latter is used in applications such as current web browsers, Stephen Kent's Privacy Enhanced Mail [Ken93] and the PKIX work [KPC]. These hierarchical structures work well within closed networks or where peer-based approaches are not suitable for management and control purposes.

However, rigid hierarchical structures do not work when artificially imposed on a system that is fundamentally peer-based, or when there is a mismatch between the semantics of the protocol underlying the imposed architecture and the system for which it is intended. For example, the global X.509 PKI architecture is currently based on the Internet naming scheme.

The structure of relationships between entities in a network must be allowed to evolve naturally. Although at its most basic, the Internet is based on peer technology (each router forwards IP packets to one of its neighbours, or broadcasts messages to all machines on a local Ethernet network), we have observed natural hierarchies emerge naturally at higher levels. Examples can be seen in PGP key introducer networks [Har] and small world [WS98] patterns in the world wide web [AJB99].

Therefore, it is essential that the model of trust that we will build is flexible and as general as possible. Consequently, our trust model will be based on a decentralised approach. This will fit in with the decentralised basic structure of the Internet and at the same time it will provide the building blocks for a stricter hierarchical trust structure, whether engineered or emerged.

Distribution has also shown its strength in many areas. For example, the attempt to create a system where extremely intelligent agents that are able to do everything is beginning to make way for an alternative paradigm, one where hundreds or even thousands of 'stupid' agents with highly specialised tasks collaborate to provide an emergent intelligent behaviour. Patti Maes, in her interview with Internet Computing[PW97], said:

> *We always think of intelligence as a centralized thing. We view even our own consciousness as centralized. It's called the homunculus metaphor – that there's a little person inside our brain running things. But it's more likely that intelligence is decentralized and distributed.*

Instead of delegating the problem of determining trust to some centralised node, like a certification authority, for correctly binding keys to names, Maes's quotation hints at an alternative distributed approach, i.e. one which involves decentralisation and collaboration. Indeed, this approach is the basis of PGP's [Zim94] technique for managing public keys. In addition, Blaze et. al has shown in [BFL96] how decentralisation improves the management of trust.

1.11 Hypothesis

Recent developments in the pervasiveness and mobility of computer systems have invalidated traditional assumptions about trust in computer communications security: there is now a gap between classic notions of systems trust that are based on certainty and control, and the more dynamic and complex environment of social trust.

In view of the fact that online agents represent users' digital identities, we believe that it is reasonable

to explore social models of trust as a means of bridging this gap. Consequently, the thesis of this work is that it is feasible to design and formalise a dynamic model of trust for secure communications based on the properties of social trust.

1.12 Summary of Contribution

The contribution of this work can be summarised as follows:

Social trust analysis. Current trust models are based on ad-hoc models of social trust and their designers' assumptions of trust's properties. Social trust itself is a vast research area spanning the areas of sociology, psychology, economics, political science, philosophy and education theory. The early part of this work investigates facets of social trust and reputation that are immediately relevant to computer and communication systems and can be used to inform their design.

Analysis of trust models. We present a rigorous survey and analysis of twenty-two computer trust models and compare them against relevant social trust properties. This contributes to the understanding of where the gaps between traditional and social trust models are and need to be bridged.

Properties of a social trust model. An outline of a trust model that is fundamentally based on social trust proposed. Several novel properties, such as different trust types and handling of trust dynamics are introduced. We view this model as one approach to instantiating social trust properties discussed in the design section.

Formal specification. A formal specification of the model is given. For relatively complex trust models, this approach helps remove ambiguities in the design. This also allows the model to be formally verified.

Algorithms. Novel contributions in terms of trust model properties include trustworthiness measures based on generalised experiences and stereotyping, recognition of the different phases of a trust relationship and its feedback loop, trustworthiness of recommenders based on consistency of recommendation quality, learning to translate between different standards of rating based on 'semantic distances' between trust values, tighter policies on evaluation of recommendation chains and a simple measure of reliability of reputational information.

Threats analysis. Threats to trust and reputation based systems is not a well researched area. In this work we attempt to contribute to this knowledge with an analysis of possible threats to such a system, the strategies available to malicious agents, the dynamics of the system in light of an attack and possible reactions by individual victims.

1.13 Publications

Publications that have resulted from this work are [ARH97a, AR97, ARH99, ARH97b] and [ARH00], with [ARH00] being the most widely referenced. A paper on the threats analysis work is under review for a conference on trust management.

In addition, the trust scale and trust evaluation algorithm presented in an early paper [ARH97a] is being used as part of an early trust model called Poblano for the peer-to-peer framework JXTA by Sun Microsystems [CY01]. Algorithms using the semantic distance concept from this work is currently being implemented in Ericsson's deepTrust project [QL04, QOD+05].

1.14 Dissertation Outline

The rest of this dissertation is organised as follows.

Chapter 2 is a survey of trust in the social sciences, looking at theories put forth by sociologists, economists, political scientists and psychologists. We then investigate the role of reputation and its properties in Chapter 3. A rigorous survey of current trust models is presented in Chapter 4. An outline of the Ntropi trust model is then described in Chapter 5. This is followed by details of each model component, formally specified in Z, in Chapter 6, Chapter 7 and Chapter 8. A simple model for the communication of recommendations, the recommendation protocol, is described in Chapter 9. Chapter 10 details recommendations for bootstrapping the model and database maintenance. This is followed by a threats analysis in Chapter 11. We described future work on the model in Chapter 12. Chapter 13 concludes this dissertation.

1.15 Chapter Summary

Global internetworking, ubiquitous connectivity and software agency are properties of current open networks that have motivated researchers and practitioners to question underlying assumptions on network security design. This has given rise to a new area of research into understanding trust, reputation and their roles in the security of computer communication. The goal of this work is to create a framework to facilitate trust computation in distributed systems. The framework we propose, called Ntropi, is fully decentralised and based on findings in the social sciences about how people trust one another.

In this chapter we discussed why trust is an important concept in security and, through a brief look at the evolution of network security, how traditional assumptions about trust has been invalidated by new practices in computing and communication.

Furthermore, the burden of decision-making in the face of uncertainty is a significant challenge for the designers of autonomous and intelligent agents. These uncertainties include the trustworthiness of other agents that must be included in collaborations and interactions.

Thus systems must now be built to take into account more sophisticated algorithms for reasoning about trust. To do this one must first understand the properties of trust. For this reason we look at trust in the social sciences.

Trust is a social phenomenon and it has been studied extensively in sociology and political science. However, one of the main findings in this area is that there is a lack of coherence in the definition of trust amongst social scientists. This points to the complexity of trust and difficulty of generalising facets of trust that were studied within isolated scenarios of social interaction.

In business and economics, reputation is seen as social capital. Furthermore, trust is seen as a 'lubricant' for effective economic transactions, for which transaction costs are seen as elements of friction.

Reputation is also seen as a mechanism for social control, allowing communities of agents to govern themselves without the need for imposed hierarchy. By cooperating and sharing information about the behaviour of other agents, 'good' and 'bad' agents can be identified and necessary strategies can be taken when interacting with them.

However, reputation is a not a foolproof method for determining trustworthiness. There is uncertainty in information because of incomplete information about recommendations made by agents and about the agents themselves. Thus, strategies for coping with uncertainties must be available. This also highlights the circular relationships between trust and uncertainty – while there is uncertainty in information used to determine trust, trust is also used to manage uncertainty through selection of trustworthy agents.

Finally, trust and reputation come into their own as effective tools for social control in decentralised networks of interacting agents. Thus the model we develop is decentralised in nature with the goal of allowing agents to autonomously discern trustworthy agents.

At the end of this chapter we described the thesis of this work, provided a summary of its contribution, highlighted some of the publications that resulted and provided an outline for the rest of this dissertation.

Chapter 2

Trust in the Social Sciences

> *Our minds have been built by selfish genes, but they have been built to be social, trustworthy and cooperative.*
>
> *Matt Ridley [Rid97]*

Before any model of trust is proposed, we must first grasp the concept and nature of trust itself. For a long time, the social sciences have provided a fertile medium for discussion of trust, and hence are an important source for us to survey and understand. However, despite the abundant trust research literature available, there is an evident lack of coherence among researchers with the definition or meaning of trust. To use McKnight and Chervany's term, there is currently a severe *homonymy* in the definitions of trust. The elusive nature of trust as well as its sensitivity to situational factors are contributors to this homonymy. Furthermore, the subject's complexity has forced many researchers to concentrate on empirical experiments that are very narrowly scoped. For example, trust has been studied in interpersonal relationships [Rot67] or in commercial relationships within a closed community in France [Har88]. Theoretical accounts also suffered from trust's deep rooted psychological and affective aspects and thus rarely completely agree. This situation paints an unsatisfactory picture because the divergent meanings of trust will make comparisons between research results difficult. McKnight and Chervany emphasised the importance of relating the divergent definitions of trust to one another [MC96]:

> *In order for researchers to make sense of the empirical and theoretical literature on trust, ways of categorizing each type of empirical and theoretical trust construct should be developed.*

Without a concrete understanding and definition of trust, any scientific work built upon it will be subject to ambiguities and inconsistent interpretations. Thus it is important that we try to capture the meanings of trust in the literature as much as possible. To guide us in our survey on the meaning of trust, we define for ourselves the following goals:

1. To group together as many definitions of trust made and if possible to unify them into a concrete workable definition for distributed computer systems.
2. To understand the role of personal and impersonal trust in social relationships.

3. To discover the properties of trust and its dynamics.

4. To identify aspects of social trust that are also relevant in distributed systems.

5. To compare the results of this survey with trust properties in existing computational trust models.

With regards to the first point of unifying the definitions of trust, we appreciate that in the social sciences generalisations or unifications may be counter-productive in terms of understanding or modelling real world diversity in social phenomena. However, we believe that in order to engineer a communication protocol where information is to be shared with minimal ambiguity in its semantics, some agreement to definitions of various data structures must be reached. This is the motivation for unifying the definitions of trust, where possible.

The areas of social science surveyed encompass sociology, psychology, philosophy (ethics), business, economics and political science. Since research work in these areas has so far been very narrowly focused due to the complex nature of trust, as mentioned above, in this survey we will attempt to study the characteristics of trust described in those specific situations and group them under more general headings. This is necessary because the goals of this survey as set out above requires us to view trust as generally as possible. Our discussion is broken down into the following sections:

- Definitions of Trust
- Trust Typology
- Representations of Trust
- Trust and Trustworthiness
- Ignorance, Mistrust and Distrust
- Trust Dynamics
- Deciding to Trust (or Distrust)
- Trusting Trust (or the importance of behaving as if we trusted)
- Miscellaneous Motivations for Trust
- Rationality and Trust

In the following sections, our discussion will concentrate mainly on the *mechanisms* of trust and material that is considered essential for a scientific and empirical treatment of trust and for engineering applicable models of it. Thus it is not an exhaustive survey of trust in the social sciences as a whole. For example, the idea of trust as a necessary building block of self-identity [Gid90] or the psychological basis of trust in maternal relationships [Mis96] are not covered here.

2.1 Definitions of Trust

As a starting point, we will review Gambetta's definition of trust [Gam88b][1]

> *trust (or, symmetrically, distrust) is a particular level of the subjective probability with which an agent will perform a particular action, both before [we] can monitor such action (or independently of [our] capacity of ever to be able to monitor it) and in a context in which it affects [our] own action.*

Gambetta's definition contains notions that are evident in other contributions to the definition of trust.

In essence, trust involves pursuing a course of action in a particular situation with the belief that the agent(s) involved (the trustees) will not disappoint the truster while they have the opportunity to do so in that particular situation.

The course of action taken may thus *result in a positive or negative outcome.* The likelihood of either outcome is reflected in Gambetta's definition as the 'probability'. The positive outcome is what is desired by the truster and what compels the truster to take that course of action. The uncertainty of the outcome is what distinguishes trust from other related constructs. The negative outcome constitutes the risk involved in the chosen action. We define *risk* here as the potential loss incurred by the truster as a result of the trustee taking advantage of the truster's trust and defecting.

Barber's first hypothesis in his theory of trust [Bar83] reflects the same notion as above. He indicates that a trusting choice is made when the truster estimates a stronger probability in the positive outcome, although the negative outcome is in some sense will incur a higher 'cost' on the truster than the benefit.

Barber's Trust Hypothesis 1 Given that Va^- is stronger than Va^+, a trusting choice will occur if:
$Va^+ \times S.P.^+ > Va^- \times S.P.^- + K.$

Va^+ and Va^- represent positive and negative outcomes respectively; $S.P.^+$ and $S.P.^-$ represent the subjective probabilities of attaining Va^+ and Va^- respectively; K represents the "security level" that the truster needs before taking action (cf. 'emotional security' in §2.2). K is subjective. A more thorough definition of each of these constructs are given in [Bar83].

In short, one takes a trusting action when the probability of success as perceived by the truster is higher than the probability of disappointment, while acknowledging that the probability of the latter is non-zero.

The idea of a subjective probability measure of likely outcomes also indicates that there exist *levels of trust.* How much we trust an agent will depend on these probabilities and the risks involved (which are related to the probabilities). Our level of trust in the potential trustee affects our course of action. We may trust one particular music chart more than another, for example, and buy new CDs based on

[1] Gambetta's definition was derived as a summary of the contributions to a symposium on trust in Cambridge, England, compiled in [Gam88b]. The volume has its foundations in areas encompassing sociology, evolutionary biology, psychology, economic theory, commerce and anthropology.

the higher-trusted chart, reflecting our trust in the music taste of the community of listeners or music buyers upon which that chart is based on.

Relating trust to probabilities may lead one to conclude erroneously that trust is some kind of prediction. The social events that occur are not well-defined repeatable experiments that lend themselves well to probability theory. Other situational factors must be taken into account when deciding to trust, such as risk and incentives for the trustee to behave as trusted. Additionally, a trustworthy behaviour in the past may not necessarily indicate future trustworthy behaviour as , again, situational factors may change. As Luhmann says [Luh79]:

> *Nor is trust a prediction, the correctness of which could be measured when the predicted event occurs and after some experience reduced to a probability value.*

In addition, trust is also *subjective* [MC96, Gam88a]. The levels of trust for each trustee are perceived differently by each truster. Similarly, the levels of risk involved in each situation are perceived differently by each truster too. Gambetta used 'subjective probability' to show this in his definition.

Any attempt at objective measurement can dangerously mislead practitioners into thinking that the value is transferable and used by another truster, which is not true for trust. If Alice trusts Bob (on matters of X)[2] and Bob trusts Cathy (on matters of X), that does not necessarily result in Alice trusting Cathy. In other words, *trust is not transitive*, which has also been formally shown in [CH96]. Furthermore, the learned experience basis of trust does not make transitivity possible in trust. As Luhmann puts it [Luh88]:

> *[Trust] is not transferable to other objects or to other people who trust.*

The subjectivity also relates to the specific situations, the context of our interaction and trustees involved. In other words, trust is subject to specific situations and trustees [Gam88a, Bar83, Luh88, MC96]. To say that one trusts another without further qualification of that statement is meaningless, unless we are relating to faith, by saying, for example, "I trust in God". Similarly unqualified trust statements makes less sense for less divine subjects, e.g. 'Alice trusts Bob'. We may trust London Underground to bring us safely from Euston to Oxford Circus, but not trust it to bring us there on time. We may trust our car mechanic to do a satisfactory repair job but not necessarily trust him to baby sit toddlers. Thus, trust is *situation* and *trustee* dependent. The question to ask is not whether Alice trusts Bob but does Alice trust Bob and for what?

The situational dependency of trust is also evident in a survey carried out by McKnight and Chervany [MC96]. In their survey, the authors picked out from the literature personal attributes that are deemed relevant to trust and clustered them into 16 categories. This large number of categories highlights the situational dependency of trust as well as the narrow scope of research that has been carried out on trust.

Another point of interest in Gambetta's definition is how trust affects "[our] own action". This point involves two important notions. The first is that of granting *control* to the truster. When we take a course of action that involves trusting another agent, the outcome of the action depends, amongst

[2]The use of 'on matters of X' is borrowed from Ed Gerck's definition of trust in his work on Meta-Certificates [Ger98]

others things, on the actions[3] of the trustee [Bai85, Das88]. This means that when we trust, there is always a chance for us to be disappointed by the trustee. However, it is the expectation that the trustee will not disappoint us, in spite of the opportunity, that characterises trust. For example, some people trust the valet service to park their cars for them, knowing that the valet parking attendants are able to just drive off with their cars. Indeed, there have been instances when the attendants have actually betrayed their trust and did drive off with the entrusted vehicles. Baier encapsulates this risk of trusting another [Bai85]:

> *Trust then, ..., is accepted vulnerability to another's possible but not expected ill will (or lack of good will) toward one.*

When we trust, we accept this risk. After the course of action has been taken we then learn whether our trust was warranted by the result of this trusting behaviour. This brings us to another aspect of trust, that is the *delay of monitoring* the actions of others until our choice of action has been made [Bai85, Gam88a, Das88]. In other words, the evidence of the trustee's trustworthiness (for our current interaction with him) is only available after trust has been granted by us. Inevitably, there will be times when even a delay is not possible and we are left to trust without ever being able to monitor the outcome. This aspect of trust is also captured by Gambetta's definition above – ".. before [we] can monitor such action (or independently of [our] capacity of ever to be able to monitor it..". Dasgupta too has observed this, as he uses the word 'trust' in the sense of [Das88]:

> *correct expectations about the* actions *of other people that have a bearing on one's own choice of action when that action must be chosen before one can* monitor *the actions of those others.*

Trust loses its potency if it is possible to undertake monitoring before the course of action is selected, as then we are in a position without risk (because we already know what the outcome will be). Unless we know how things will turn out, we will need to place trust in others. This also points to the *future orientation* of trust. This is usually displayed as *expectations* about the future [Bar83, Das88, Gam88a, MC96]. Barber, for example, says that trust implies expectations of the future, which can be broken down into three forms of expectations [Bar83]:

1. Expectation to fulfil moral orders.
2. Expectation to perform roles competently.
3. Expectation that fiduciary obligations will be met.

At this juncture, we can summarise the properties of trust as follows:

1. Trust is subjective.
2. Trust is situation specific.

[3]Note that the target of a trusting behaviour is an 'active' entity, one which has the ability to perform action independent from the truster's control.

3. Trust is agent specific.
4. Trust is not absolute, it exists as levels of trust.
5. Trust involves expectations of future outcome.
6. Situations of trust can result in positive or negative outcomes, thus involves risk, uncertainty and ignorance.
7. Trust gives control to the trustee and an opportunity to betray the truster.
8. The inability to verify one's actions until after the action has completed requires trust in the trustee prior to the action being taken.
9. Trustees are active agents that have the ability to perform with a degree of independence from the truster's control.
10. Trust is not a prediction.
11. Trust is not transitive.

Next, we compare trust with other related notions that are sometimes used interchangeably with 'trust'.

2.1.1 Trust and Confidence

The difference between trust and *confidence* is that trust involves choosing between alternatives whilst confidence does not [Mis96, Luh88]. When you choose an option among alternatives, in spite of the risk of being disappointed, you trust. Confidence has also been described as a *habitual* expectation, e.g. "I am habitually confident that my milkman will deliver milk to the doorstep tomorrow" [Mis96]. Misztal explains [Mis96]:

> *The main difference between trust and confidence is connected with the degree of certainty that we attach to our expectations. It is, of course, much easier to decide whether to have confidence or not in one's milkman than to decide which people can be trusted to reciprocate friendly actions, since trust involves a difficult task of assessment of other people's capacities for the action.*

There is a recursive relationship between confidence and trust [Luh79], an increase in one forms a stronger foundation for the other. When faced with situations of a certain degree of uncertainty, trust is required in making decisions. At the same time, trust can only be extended when there is a certain level of confidence already present, in order not to trust 'blindly' [Luh79]. Even on a personal level, the person possessing stronger self-confidence is more ready to trust because self-confidence allows us to better accept unexpected problems and makes any insecurity 'bearable'.

2.1.2 Trust and Reliance

Reliance on something or someone is not necessarily an indication of trust [Adl94]. We may rely because we have to, or because sometimes it is the best or only choice available to us. An example

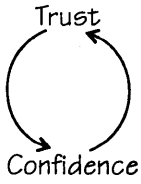


Figure 2.1: Trust and confidence have a recursive relationship.

is when we ask a stranger for directions. The person giving directions may have, on purpose or not, misled us by giving wrong directions, but, in this case it is a choice between relying on uncertain information, or no information at all.

However, this hints at some underlying disposition to trust some abstraction of basic benevolence in all humans, for otherwise we wouldn't ask the stranger for instructions as no information would otherwise be worth relying on. As we will see in §2.4.4 a trusting disposition is an important component of trust based decision making. Baier [Bai85] gives her account of reliance thus:

> *We can still rely where we no longer trust. What is the difference between trusting others and merely relying on them? It seems to be reliance on their good will toward one, as distinct from their dependable habits, or only on their dependably exhibited fear, anger, or other motives compatible with ill will toward one, or on motives not directed on one at all.*

2.1.3 Trust and Hope

Trust also differs from *hope* in terms of available choices. When a potentially risky action has to be taken, we hope that it will result in something satisfactory. Trust relinquishes its role here as there is now no choice to be made.

2.1.4 A Note on Defining Trust

At this point, it is an easier task to produce the properties of trust than to define exactly what trust itself is. The reason for this is that trust involves a combination of interrelated cognitive and non-cognitive constructs, some of which may or may not be called on depending on the entities and situations involved. Hence, McKnight and Chervany proposes that trust be "characterised as a set of inter-related constructs' [MC96]. In order to fully comprehend the constructs they defined, further understanding of the properties and types of trust is required. Thus we will postpone this discussion and look first at the different types of trust.

2.2 Typology

In the literature, two kinds of trust typology may be found. The first relates to what we trust *in*. Barber's three types of trust [Bar83] falls into this category:

1. General trust in moral social order.

2. Specific trust in technical competency.

3. Specific trust in fiduciary obligations.

Although this kind of typology is useful, it is not general enough for our purposes. It merely describes the structures in which one would expect to find trustworthy behaviour and falls short of the ability to give insight into how trust itself actually works. What we are more interested in is a second kind of trust typology, which describes the different types of trust that may be present in any single situation. The difference between the types relates to the subject of trust, or for whom or what the trust is granted. From here on we will use the term 'trust typology' to refer to this second kind of typology.

Luhmann says that the role of trust is to handle the complexities of modern life, and thus encompasses our personality as well as formal social systems [Luh79]. In any social interaction, any two or all three of these entities are involved – ourselves, the agent we are interacting with and our environment. For example, our relationship with the members of our family is a personal one and involves us and the other family member, whereas in a business relationship, it may involve the business partners as well as the legal environment within which contracts between the business partners are maintained. In the latter case, the judicial system upholding the contract law of the country may be seen as the third party in the business relationship. This gives us three types of trust [MC96]:

1. System trust (or Impersonal/Structural trust)

2. Dispositional trust

3. Interpersonal trust

> **Definition 1 (Trust Relationship)** *A trust relationship exists when an entity has an opinion about another entity's trustworthiness. Thus, trust relationships do not exist between strangers or an entity that has no knowledge about another's existence.*

System trust refers to a trust relation that is not based directly on any property or state of the trustee but rather on the property of the system or institution within which the trust relation exists. For example, we trust the judicial system to uphold contract law, and thus we trust, albeit in an indirect fashion, the parties in the contract. Much current work in computer and communications security is geared towards increasing the system trust of computer networks. For example, by formally proving that an encryption algorithm meets certain desirable properties, it can give users of this encryption algorithm more confidence in its ability to protect their communications – the users have system trust in the cryptosystem, which may lead to system trust in the communication network which implements that cryptosystem. System trust is situation specific.

McKnight and Chervany [MC96] identifies two types of impersonal structures that can be the subject of system trust, that is *structural assurances*, like regulations, contracts and cryptosystems, and *structural normality*, that may include the truster's or trustee's roles in the situation. Structural assurances seek to 'reassure' us that measures have been taken specifically to safeguard us and reduce the risk of something going wrong. Structural normality refers to the appearance that everything is

'normal'. This gives the truster a sense of security, which positively supports his trusting decisions by increasing his tolerance for perceived risks.

System trust, according to Luhmann, has gained predominance in modern social relationships over the more spontaneous interpersonal trust. There is a reliance on the system to maintain conditions and to perform satisfactorily, rather than trusting the other person. It also involves some amount of reflexivity and a conscious approach to trust – as long as there is a belief that everyone cooperates and trusts, then there will be trust [Luh79].

Dispositional trust, sometimes referred to as 'basic trust', describes the general trusting attitude of the truster. This is "a sense of basic trust, which is a pervasive attitude toward oneself and the world" [MC96]. Therefore it is independent of any specific party or situation. A person's trust disposition decides how much initial trust to give and also affects how the truster react to feedback from interactions that affect trust [BF87, Rot67]. This trust disposition, according to Boon and Holmes [BH91], is a major part of who we are and is rooted deeply in childhood experiences. Dispositional trust is cross-situational.

McKnight et al., in [MCC95], breaks down dispositional trust into two further subtypes. *Type A* dispositional trust concerns the truster's belief of whether other people are generally good, trustworthy, and therefore should be trusted accordingly. *Type B* dispositional trust is a more 'active' construct because it concerns the truster's belief that irrespective of whether others are good or not, a more positive outcome can be obtained by acting 'as if' we trust the other person (see §2.8.4).

Interpersonal trust is a trust relation based directly on the properties of the trustee. In other words, the behaviour of the trustee directly affects the truster's trust towards him. Interpersonal trust is situation specific.

Trust types	Situational	Cross-situational
Personal/Interpersonal	×	
System	×	
Dispositional		×

Table 2.1: Trust types.

In addition to the three types of trust above, there is also the concept of *blind trust*. Blind trust is not another trust type, but can be seen as a subtype of each of the three trust types above. It exists as extreme levels of trust for any of types above [Luh88]. Even when contradictory evidence is present, trust is still maintained regardless. This is explained by Gambetta [Gam88a]:

> *.. blind trust or distrust represent lexicographic predispositions to assign the extreme values of the probability and maintain them unconditionally over and above the evidence.*

2.3 Describing Trust as a Set of Related Constructs

We will now look at McKnight and Chervany's characterisation of trust as a set of inter-related constructs [MC96]. Figure 2.2 below shows the constructs and their relationships.

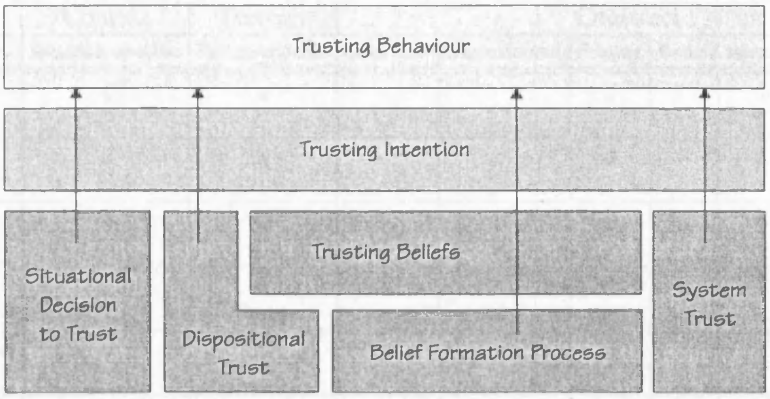


Figure 2.2: McKnight and Chervany's trust constructs. Arrows show relationships – constructs crossed by arrows are those that mediate the relationship.

This set of constructs does not fully cover every type of trust discussed in the literature but is intended to represent what the authors considered to be the most important forms of trust. The relationship between the constructs is as follows, as explained by the authors:

> *.. beliefs/attitudes (in our case Trusting Beliefs) lead to intentions (Trusting Intention), which, in turn, readily become manifest in behaviours (Trusting Behaviour). The logic here is simple. When one has Trusting Beliefs about another, one will be willing to depend on that person (Trusting Intention). If one intends to depend on the person, then one will behave in ways that manifest that intention to depend (Trusting Behaviour).*

A *Trusting Intention* is defined as "the extent to which one party is willing to depend on the other party in a given situation with a feeling of relative security, even though negative consequences are possible" [MC96]. The 'feeling of relative security' is an emotional aspect of this construct, and this, according to the literature, is what allows the truster to take that 'leap of faith' when deciding to trust.

Trusting Behaviour indicates the act of trusting, based on Trusting Intentions. Trusting Behaviour is a behavioural construct while Trusting Intention is a cognitive-based construct.

Trusting Beliefs is the truster's belief in the level of the trustee's trustworthiness as well as the truster's own confidence in that belief. The four most prevalent trust-related beliefs in the literature, as surveyed by McKnight and Chervany [MC96] are a) *benevolence*, b) *honesty*, c) *competence* and d) *predictability*. Trusters evaluate potential trustees mainly on these properties before making a trust decision.

Situational Decision to Trust is the truster's willingness to trust for a given situation *regardless of the entities involved*. This may arise when the truster regards particular situations as very beneficial with low risk, irrespective of the perceived trustworthiness of the parties involved in that situation.

Dispositional and *System Trust* have been discussed in the previous section. The distinguishing dimensions of the six related trust constructs are reproduced here from [MC96] in Table 2.2.

	Context	Persons	Construct					
Trust Type	Situation-specific	Person-specific	Structural	Dispositional	Feeling	Belief	Intention	Behaviour
Trusting Intention	•	•			•		•	
Trusting Behaviour	•	•			•			•
Trusting Beliefs	•	•			•	•		
System Trust	•		•			•		
Dispositional Trust				•				
Situational Decision to Trust	•						•	

Table 2.2: Properties of McKnight and Chervany's six trust constructs [MC96].

2.4 Representations

In this section we will look at how trust is represented, e.g. as a belief or some probability distribution. This is an area of much debate in the area of computer security because although as humans we are adept at determining the optimal level of trust to grant in any given situation, we are not predisposed to putting the value or level of that trust down on paper as an exact measurement or unit. Therefore, implementations of trust 'values' usually result in ambiguous representations [RS97]. Nevertheless, in any given situation, we are able to say how beneficial it is to us to trust, or distrust. There is value in trusting and there is value in being trustworthy. Dasgupta parallels this 'value' of trust with knowledge and information [Das88].

> *.. even though there are no obvious units in which trust can be measured, this does not matter, because in any given context you can measure its value, its worthwhile-ness. In this respect, trust is not dissimilar to commodities such as knowledge or information.*

If we were to model trust for computer systems, such trust values must be represented in some form or another.

2.4.1 The Trust Relation

Trust is a three-part relation [Har93, Bai85, Luh79]:

> *A trusts B about X*

where A and B are entities and X is the specific situation or action that A trusts B for. For example Alice trusts Bob to invest $1000.00 of Alice's money competently without running off with it. In this case, the variables A, B and X are replaced with Alice, Bob and 'to invest $1000.00 of Alice's money without running off with it' respectively.

2.4.2 Trust as Subjective Probability

Trust has been described as a *subjective probability* value ranging from 0 to 1 inclusive [Gam88a]. Meanings for three of the possible values were given by Gambetta as shown in Table 2.3.

Value	Meaning
0	Complete Distrust
0.5	Uncertainty
1	Complete Trust

Table 2.3: Gambetta's trust value semantics.

The values, suggests Gambetta, are spread across a "probabilistic distribution of more general expectations". However, Gambetta does not go into more detail than that. He merely suggests this scheme intuitively, and possibly as a guide towards a more concrete explanation of trust. Since trust is subjective, it is also possible that different forms of distribution exists. If we were to use Gambetta's trust values above as a guideline, the graph of trust values can be plotted in many ways for each truster and for each situation, some of which are shown in Figure 2.3.

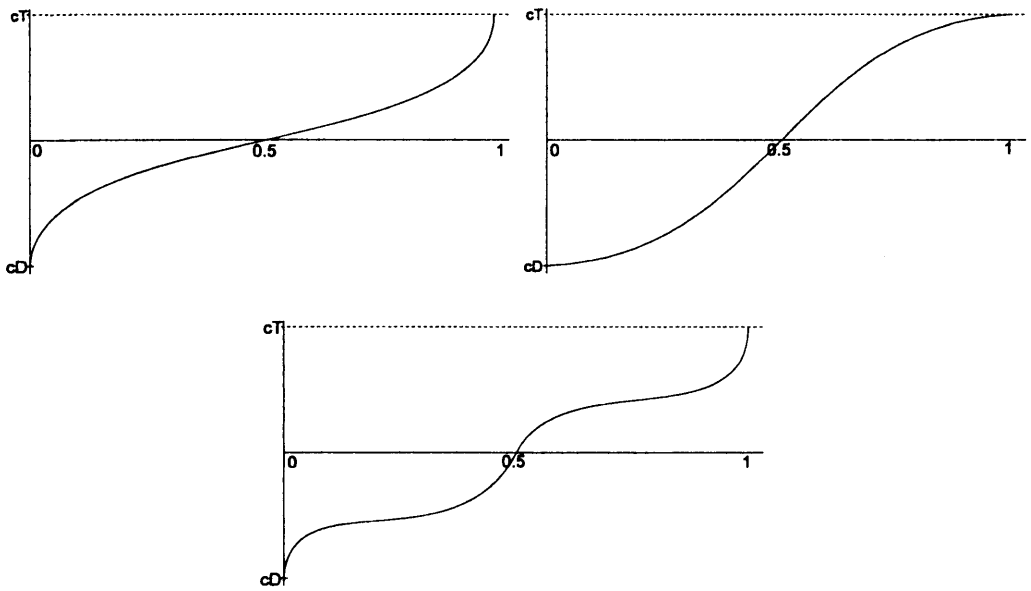


Figure 2.3: Some possible trust value distributions for each truster in each situation, with values between complete trust (cT) and complete distrust (cD). These are just illustrations of what's possible - indeed, the distribution may not even be linear, unlike the graphs shown above.

This representation of trust values is used in theoretical trust models in computer science and information security research (see Table 4.7, page 97). The reason is simple – it is an intuitive representation. Some variations extend the scale to include -1 as the value for complete distrust rather than 0.

The exact semantics of Gambetta's 'complete distrust' (0) and 'complete trust' (1) were not given. Without further elaboration, these values are ambiguous and misleading. Consider a trustee, Bob, whom Alice presumes to have a trust value of 1, with respect to a certain situation or context X. Does this mean that:

1. Alice thinks Bob *is completely trustworthy*, or
2. Alice has *complete trust in* Bob?
3. Is there a difference between (1) and (2)?

Furthermore, the use of 1 or 0 indicates a 'maximum' value. Thus, in the example above, does it mean that there are no entities more trustworthy than Bob? The source of this ambiguity can be traced to Alice's internal representation of her 'trust database' of entities and their trust values.

It is clear that, although Gambetta-like trust value representations are intuitive, they would be naive representations for use in computer systems. Without a clear semantics and concrete 'data structure', such trust value scales are not very useful.

2.4.3 Trust as Threshold

The notion of trust as a *threshold* point has also been suggested [Luh79, Gam88a]. This threshold point defines the point at which one decides to trust or not to trust[4]. Luhmann explains that since the role of trust is to reduce the complexities of life, this threshold provides a simple mechanism for trust, which is 'easy to master', and thus serves to lessen complexity in decision making. Using the same representation as above, Gambetta [Gam88a] says that trust is better seen as

> *... a threshold point, located on a probabilistic distribution of more general expectations, which can take a number of values suspended between complete distrust (0) and complete trust (1), and which is centered around a mid-point (0.50) of uncertainty.*

2.4.4 Dispositional Trust

Hardin gives a Bayesian account of learned trust in [Har93]. The model of trust he presented represents the levels of dispositional trust, rather than trust that is granted in arbitrary situations for arbitrary trustees. The basis of his model is an iterative re-evaluation process of a person's trusting disposition through experiences. As this concerns the dynamics of learned trust, the process itself will be discussed in depth in §2.7.4. We show how the model is represented here.

In his trust model, Hardin assumes that there is an objective real-world where the distribution of trust is linear from 0% trustworthy to 100% trustworthy. When trust is not disappointed (i.e. the truster is not betrayed), there is a positive amount of payoff, and there is negative payoff when trust is defaulted on. There is also an objective break-even point "at which the average return from trusting a person of that degree of trustworthiness is neither gain nor loss" [Har93]. This point occurs at the intersection of the trust graph and the zero gain/loss line (at H, T or L). Higher trusters perceive this point is reached for people with relatively lower levels of trustworthiness (H) while low trusters only perceive this is reached for very trustworthy people (L). Optimal trusters perceives the break-even point to be where it really is in the objective real-world (T).

The trust value is the subjective probability of getting an objective amount of expected gain. In other words, the actual gain is an objective value commonly known to all entities in an interaction (e.g.

[4]Note that to 'not trust' does not equate to 'distrust'.

"a hundred pounds Sterling") while the probability of obtaining that gain differs from one truster to another. The graph showing this for different trusting dispositions is shown in Figure 2.4 below.

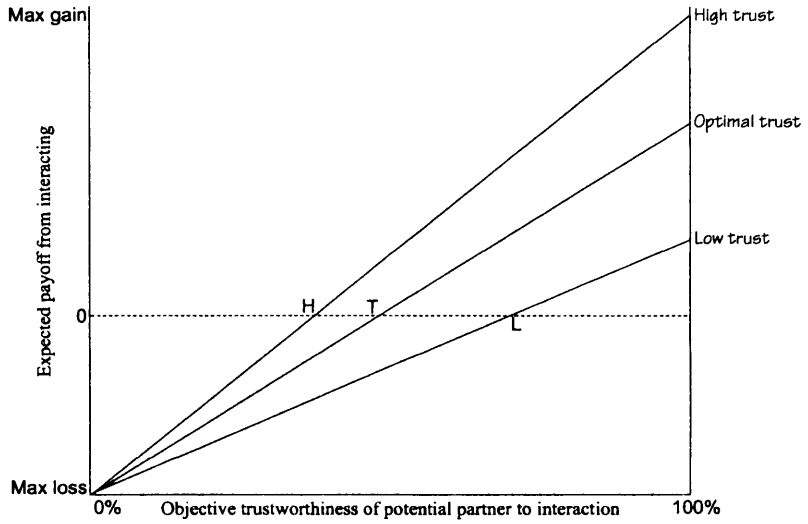


Figure 2.4: Hardin's trust model.

Again, Hardin's model represents only dispositional trust, which is constantly being re-evaluated in light of new experiences and learning. With each new experience, the graph of the high and low truster moves closer to the graph of the optimal truster.

In practice, the model presented by Hardin above is rather simplistic, and Hardin himself admits this. He outlined six shortcomings of the model [Har93]:

1. The model ignores the relative size of loss and gain at risk.
2. Different situations that call for different levels of trust to be granted for the same trustee are not accommodated.
3. Strategic decisions, such as 'as-if' trusting, and trusting incentives, such as repeated interaction with the same person, are not handled by the model. (Thus this model represents McKnight and Chervany's Type A dispositional trust, as described in §2.2).
4. The model is only half strategic – it only assumes strategic calculations by the potential truster and does not attribute any sophistication to the potential trustee.
5. The complexity of possible ways of learning is ignored by the model.
6. The model may not allow certain skews in trusting people, e.g. the automatic trust people put in trustees of a certain professional level.

We further note that the convergence of the three trust types at the origin may not be realistic as the perception of the potential partner's trustworthiness may be higher or lower than this based on the disposition of the truster. Furthermore, the assumption of linearity in Hardin's graph may not be realistic, as we have shown in Figure 2.3.

2.4.5 Trust as Belief

As we discussed earlier, trust is subjective (§2.1). Thus any representation of trust, e.g. like those just described above, are also used subjectively. In other words, any level or value of trust only exists as a subjective belief rather than an objective property of some entity or relationship between two entities. Indeed, a large percentage of the literature represents trust in terms of expectations or belief [MC96]. In this work, we use the term *belief* as follows:

Definition 2 (Belief) *A belief is an agent's acceptance of something as truth.*

So when we say that Alice believes Bob is guilty, to Alice, it is true that Bob is guilty, while others may believe otherwise.

2.5 Trust and Trustworthiness

Trustworthiness is seen more as a property of another agent while trust involves the temporal 'relationship' between the truster and trustee. An agent's reputation for trustworthiness may not necessarily be the precondition for the granting of trust to him as it is "possible to claim on the one hand that a person is untrustworthy and on the other hand that he can be trusted to do what he said he would on a given occasion. This is because on this occasion he may have the right incentive." [Das88]. For example if a seasoned armed robber warns that he will kill you if you 'try anything' while he empties the vault, it may be foolish to trust that he will not. Dasgupta draws the distinction between trust and trustworthiness [Das88]:

> *'Trustworthiness' concentrates on a person's overall disposition, his motivation, the extent to which he awards importance to his own honesty. Being able to trust a person to do what he said he would, on the other hand, requires us to know not only something of his disposition, but also something of the circumstances surrounding the occasion at hand. If the incentives are 'right', even a trustworthy person can be relied upon to be untrustworthy. 'Every man has his price': repugnant though it is to our sensibilities, the cliché captures the view that no one awards an infinite weight to his own honesty.*

Definition 3 (Trustworthiness) *An agent's trustworthiness is his reputation for being worthy of a certain level of trust in a given situation.*

2.6 Ignorance, Mistrust and Distrust

Thus far we have used trust in both its positive (trust) and negative (not trust) notion. In this section we will be more specific and briefly look at what it means to distrust, to have mistrusted or just to be ignorant of another's trustworthiness.

In this work, we will use *distrust* according to the following definition:

Definition 4 (Distrust) *To distrust is to take an action as if the other agent is not*

trusted, with respect to a certain situation or context. To distrust is different from not having any opinion at all on whether to trust or not to trust, i.e. ignorance.

Although distrust is a 'negative' form of trust, it is not the negation of trust. If it was then we would not have a choice between trust and distrust as to determine one we would just need to reverse the sign of the other. Distrust, as Luhmann says, is *functionally equivalent* to trust [Luh79].

To understand this concept, it is necessary to understand Luhmann's position that the function of trust is the reduction of social complexity. Trust "simplifies life by the taking of a risk" [Luh79]. When a person is not ready to trust, he becomes overburdened with complexities whose surplus can render him ineffective or incapable of action. Thus a functionally equivalent strategy must be called on to reduce complexity and make the situation more tractable. Therefore an approach that is 'negative' to trust is used and he calls on distrust. Distrust, then, is trust's functional equivalent in its complexity reduction role.

Distrust can be provoked by the sudden appearance of inconsistencies, like sudden strange noises occurring in quiet of the night. Distrust can also be induced by 'symbolically discrediting symptoms', e.g. when a person we trust behaves contrary to what we perceive should be trustworthy behaviour. We will refer to this latter situation as one of *mistrust*, as in the definition below:

Definition 5 (Mistrust) *When a trustee betrays the trust of the truster, or, in other words, defaults on trust, we will say that a situation of mistrust has occurred, or that the truster has mistrusted the trustee in that situation.*

Since the function of trust and distrust are to minimise the complexity, Luhmann argues that there exists in each of us a *threshold* that serves to simplify the task of determining a transition from trust to distrust. The level of trust is kept at a level, despite accumulated experiences, until the threshold is reached; here a small step can make drastic changes – the transition from trust to distrust. This is why not all experiences raise doubts, rather they are kept 'latent' beneath the threshold until that 'incalculable moment' [Luh79].

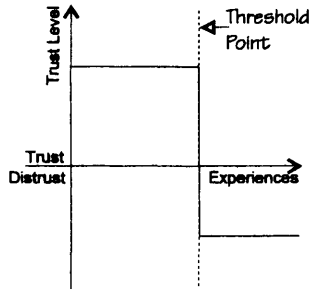


Figure 2.5: Transition from trust to distrust at the threshold.

The danger of this is that in social interactions, distrust tends to be self-reinforcing, which, according to Luhmann's system-theoretic view of trust, is based on a principle of feedback:

a wrongly or insecurely adapted system brings itself into equilibrium with its environment not by correcting itself on the basis of its effects but by finding its effects endorsed and hence offering occasions for new causes.

Lastly, an absence of trust should not be equated to distrust. In between trust and distrust is where we find a situation of *ignorance*. This is where there exists "various forms of relying on and taking for granted which are not grounded in either optimism or pessimism about the other's goodwill" [Jon96]. We may also add that when the truster is being ignorant, he has no opinion about the other person's trustworthiness. The action that the truster consequently chooses will thus be based on other factors or his trusting disposition. In this work, we define ignorance thus:

> **Definition 6 (Ignorance)** *We say that a truster is 'ignorant' of a potential trustee when he holds no opinion about the potential trustee's trustworthiness.*

2.7 Trust Dynamics

Once a trust relationship is established, the truster's belief about the trustee's trustworthiness is not static and changes with experience and time. In other words, trust is dynamic. As noted by McKnight et al. [MCC95], the literature contains statements that describe trust as, amongst other things, fragile or easily destroyed and as something that takes time to form or forms very quickly.

Experience plays a vital role in the dynamics of trust. Trust cannot be willed nor imposed [Bai85], but must be learned [Har93] through repeated encounters and experience [Das88, Gam88a, Luh88]. As Misztal puts it, "due to its fragility, trust can be learned only very gradually on the basis of previous experience" [Mis96].

Luhmann stresses the importance of experience indirectly by using a related notion – familiarity. Familiarity comes about through experience and "familiarity is the precondition for trust as well as distrust" [Luh79]. According to Luhmann, familiarity plays a much more important role than truth or confidence. In fact, he states that neither truth nor confidence are even relevant while familiarity, which involves time and learning, is. The role of experience in trust is twofold:

1. Aggregated, or generalised, experiences form the basis for new trust relationships.
2. Individual experiences change the level of trustworthiness of the trustee, as perceived by the truster.

For online interactions, particularly consumers' interactions with e-commerce sites, research has shown that the user's (or consumer's) trust relationship with the e-commerce site goes through various phases [RA99]. Their findings showed that the relationship started with users being *unaware* of the site followed by a period where the user starts *building* some amount of trust towards the site. The user then looks for evidence or cues to *confirm* his trust and once confirmed further experience is required to *maintain* the trust relationship.

In summary, there are three main phases that a trust relationship exist in:

- **Unfamiliar** phase - where the truster is unaware or unfamiliar with the prospective trustee.
- **Formation** phase - where the relationship is being built.
- **Stable** phase - where the bond between truster and trustee is strong and require evidence only to maintain it.

2.7.1 Forming a New Trust Relationship

In first-time interactions, there is no experience with the potential trustee yet with which to form an opinion about its trustworthiness. In Hardin's model, trust towards the new person is based on the trusting disposition of the truster. This trust is a generalisation of past experiences with trust, but not directed towards the new person specifically[5]:

> *My prior experiences with trust may have been so charmed that I optimistically trust this new person. Or they may have been so disastrous that I pessimistically distrust her or him. The new person is no different in the two cases; my alternative experiences, unrelated to this person, are the source of difference. Experience moulds the psychology of trust [Har93].*

For example, in a particular town which we will call Recklessville, where motorists have general disregard for traffic regulations, a pedestrian, Alice, crossing the road at pedestrian crossings may have had near accident experiences when motorists refuse to stop or slow down. These experiences may generalise in such a way that Alice learns to give way to traffic before crossing the road, even at pedestrian crossings. On a trip to another town, Safetown, which has generally rule-abiding motorists, she will still have the same distrust for motorists at pedestrian crossings and will continue to give way to traffic, until she learns that in Safetown motorists can be trusted not to run you over when you cross, perhaps by repeated observation and by following the examples of others.

We can also say that Alice's aggregated experiences for a particular situation form a generalised expectancy for that situation. She expected motorists in Safetown to run people over like they do in her home town Recklessville but only later learns that Safetown's motorists are much more considerate and rule-abiding.

Further support for the role of dispositional trust in forming new trust relationships are given by McKnight et al. in [MCC95], and also describes the effects of the two types of dispositional trust, as discussed in §2.2. The discussion so far on the role of dispositional trust in first-time encounters is relevant to McKnight et al.'s Type A dispositional trust. For Type A, in the new trust relationship, the truster's trust belief is directed at the trustee. Type B dispositional trust on the other hand "influences one to trust the other without respect to beliefs in the other person" [MCC95]. Type B reflects a more positive and cooperative attitude on the truster's part, which has no relation to his belief of whether the trustee is genuinely trustworthy or not.

Additionally, McKnight et al. also identified other factors which influence the formation of new trust relationships. The complete list is as follows [MCC95]:

1. Dispositional trust, both types A and B (see §2.2).
2. Categorisation and illusion.
3. System trust (see §2.2).
4. Trusting beliefs.

[5] We will ignore in this discussion 'first-impression' information about the new person that can also be used to make trust-related judgements, unless where its inclusion is specifically required.

Categorisation and *illusion* are both described by McKnight et al. as belief-affecting trust enabling mechanisms. In order to handle the complexities of everyday encounters, people tend to categorise groups of information in such a way that equivalent responses to members of the same category can be given. For new trust relationships, three categorisation mechanisms are used:

Unit Grouping With unit grouping, the truster places himself and the trustee(s) into a new grouping. Those in this new grouping share common goals and/or beliefs, values and assumptions. This cognitive grouping produces a sense of security that is conducive to trust. "This feeling of security, combined with the cognitive beliefs, forms a basis for interpersonal trust" [MCC95].

Reputation Categorisation The reputation of a person, individually or as part of a reputable group, also influences initial new trust relationships. "The person may be perceived as a competent person because of his/her own actions, or because s/he is a member of a competent group" [MCC95]. Dasgupta also gave an illustration of this by citing the example that individual Ghurkas, although varying in levels of courage, are still thought of as being very courageous simply because they are Ghurkas: "he is one of them" [Das88].

Stereotyping General biases in opinions on various levels (e.g. gender or specific groups) form prejudices. This stereotyping, combined with first-impressions obtained from physical traits or word-of-mouth affects the truster's beliefs about the potential truster, and consequently whether to trust the potential trustee or not.

These categorisation mechanisms affect the truster's beliefs about the new person or entity. When there is insufficient information for a logical categorisation process to take place, *illusionary mechanisms* are used [MCC95]. With this mechanism, the lack of information to make trust judgements with are compensated with made-up information. This made up information is usually optimistically formed such that situations of utility to the potential truster can be entered. However, they can be either high or low depending on the truster's disposition.

Truth is not essential to trust [Luh88, Das88], which is why such illusionary measures can be effective. This is especially true for system trust where the task of scrutinising the complexity of the system prior to making trust decisions is too large. As Misztal illustrates [Mis96]

> *For example we have confidence in the purchasing power of money, which is based more on our familiarity with it rather than on our precise knowledge of the working of the currency. Moreover, we are not involved in the process of deciding whether or not to accept money each time we encounter it. We assign our expectations to it, knowing – on the basis of our previous experience – that the danger of disappointment is minimal. This form of trust, which neglects the possibility of disappointment, is learned as behaviour and involves a feeling of familiarity. It covers the risk involved and the reflexivity, initially necessary for these relationships.*

If proper impersonal structures are in place, initial trust can be high, irrespective of whether the prospective trustee is trustworthy or not. This new trust relationship is based on *system trust* as described in §2.2. System trust may be based on the appearance that things are "normal", on "structural safeguards" [MCC95] or on the specific situation itself. A formal contract between new business

partners is an example where the initial trust relationship is based on system trust, i.e. trust in the judicial system to uphold contract law.

Trusting beliefs contain cognitive and emotional aspects. The truster's belief in the potential trustee's trustworthiness affects the truster's cognitive willingness to depend on the potential trustee. This is the cognitive aspect. The emotional aspect relates to the emotional security of the truster. This emotional security affects the truster in two ways – the feeling of security about the willingness to depend on another, and the feeling of security about the belief in the potential trustee's trustworthiness.

The foundations of initial trust above are weak and laden with rough assumptions, hence the highly fragile nature of initial trust. By fragile we mean the ease with which the level of trust can spiral downwards towards distrust. McKnight et al. explain in [MCC95] how each of these bases can be destroyed. An initial trust which is based on a larger number and diversity of these factors will be stronger. Gambetta says that trust is belief based on lack of contrary evidence. This again shows trust's fragility as it is usually easier to uncover evidence of untrustworthiness then trustworthy behaviour [Gam88a]. Since trust also depends on familiarity, as Luhmann states, any sudden appearance of inconsistencies and perceived "symbolically discrediting symptoms" can give rise to distrust.

In some situations, small amounts of doubt about the trustee's integrity or competence will still allow the truster to continue trusting, until a critical level of uncertainty is reached. McKnight et al. recommend that trusters in a new trust relationship adopt a 'trust-until' approach when encountering new situations or potential trustees. Since early trust is based more on assumptions than fact, care should be taken in order to prevent oneself from being the subject of opportunistic behaviour.

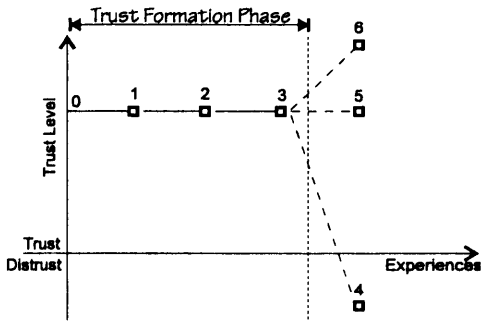


Figure 2.6: Trust level after each interaction. Dotted lines are example possible Trust levels after leaving the trust formation phase. After initial trust given (0), it may take several 'trial experiences' in the trust formation phase (1,2 and 3) before having a more stable trust relationship (5 or 6). Alternatively, if there had been a negative experience, e.g. at (3), trust may break down into distrust and no further trust relationship is maintained (4).

2.7.2 Revising Established Trust

Trust is maintained based on experience. We recall Baier's suggestion that to trust is to postpone accounting. Once trust is given, and the outcome of that trusted interaction received, accounting is carried out and this is when trust is appraised. Negative experiences reduce the level of trust while positive ones increase or maintain it. Boon and Holmes stresses the importance of experience, or more accurately, 'history' to trust [BH91]:

The particular history of a relationship may be considered a contextual variable of fundamental importance as it imparts a refined and perhaps unique quality to the expectations those involved possess about each other.

During this phase, accounting will be more narrowly scoped and trust is only appraised for the relevant trustee for relevant situations, unlike dispositional trust, which, although also dependent on experience, is cross-situational and non-personal. So, for example, we may initially have low trust for a recommended car mechanic when forming a new trust relationship with him. After two or three experiences with him we may be confident in his expertise and trust his honesty and thus complete the fragile trust formation phase and start a more stable trust relationship. When we are past the trust formation phase trust becomes less fragile and negative experiences have softer impact now than they have in the formation phase.

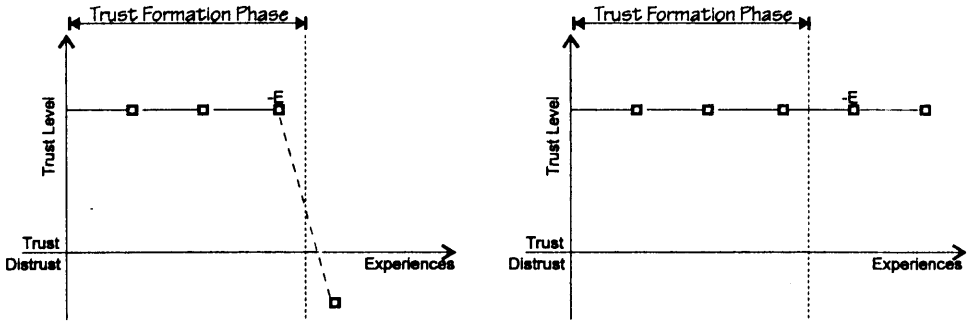


Figure 2.7: Similar negative experiences (-E) impact trust differently during (left) and after (right) the fragile trust formation phase.

McKnight and Chervany identified four major beliefs about the trustee that induce a revision in our trust for others [MC96, MCC95]. They are:

1. **Benevolence.** Is the trustee acting with the welfare of the truster in mind?
2. **Honesty.** Has the trustee been truthful and have his promises been kept?
3. **Competence.** Has he behaved competently towards the task at hand?
4. **Predictability.** Are his actions consistent enough to forecast outcomes in future interactions with him?

In our search for evidence of trustworthy or untrustworthy behaviour, one must take note of Baier's caution [Bai85]. The fragility of trust must be taken into account when querying evidence because a sudden lapse of tact on the truster's part may be perceived as untrusting behaviour by the trustee. Asking an acquaintance, for example, about whether another friend has been round to feed the hamster while one was away, may be damaging to our relationship with that hamster-feeder friend if word was to get round to her that we have been monitoring her without her knowledge. As Baier puts it:

Trust is a fragile plant, which may not endure inspection of its roots, even when they were, before the inspection, quite healthy.

2.7.3 Breakdown of System Trust

System trust relies on the fact that "others also trust"[Luh88]. This can be looked at from two different points of view – as trust in the system from the truster and from the trustee. The first is perhaps the more obvious. If we do not trust in the system to maintain conditions favourable to trust, we will lose our trust in its ability to prevent our trustees from betraying us. As Dasgupta puts it [Das88]:

> *... trust among persons and agencies is interconnected. If your trust in the enforcement agency falters, you will not trust persons to fulfil their terms of an agreement and thus will not enter into that agreement. By the same token, you will not trust the enforcement agency – for example, the government – to do on balance what is expected of it.*

Secondly, we may feel that the trustee does not trust the system. Extending Dasgupta's example above, we may feel that the trustee does not believe the enforcement agency will do what it is expected to do. This may, in turn, reduce our confidence in the agency, perhaps also with the suspicion that our trustee knows something about the agency that we do not. Furthermore, it may give us the insecurity that, since the trustee does not trust the agency, he may be more willing to default on our agreement, or betray our trust.

2.7.4 Revising Dispositional Trust

As in our trust targeted at others, our trusting dispositions too are dynamic. They change with each experience and, with each new experience, our trusting disposition gets closer to what Hardin calls in his model the 'optimal' trust level. However, the speed at which our dispositions race towards the optimal level and the risks we can potentially encounter in the process depend on whether we are optimists or pessimists when it comes to trust. Below we reproduce Hardin's trust model as in Figure 2.4, this time with the expected payoffs included.

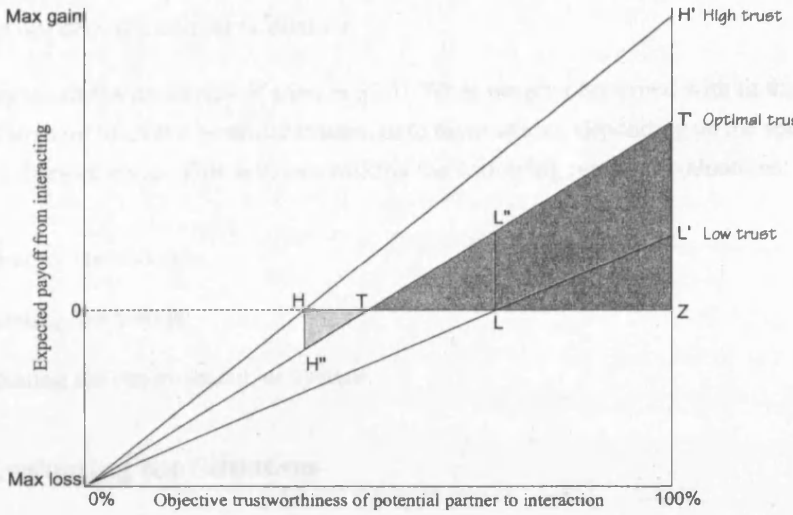


Figure 2.8: Hardin's trust model with payoffs and losses.

To refresh, the horizontal line at 0 in Figure 2.8 represents the break-even point at which the average return of trusting someone with that level of trustworthiness is zero (i.e. no gain nor loss). For all

three types of truster, high, optimal or low, trust will only be granted above the break-even point. Thus, the high truster trusts most frequently (in the range H to Z) and the low truster trusts least frequently (between L and Z). This is because the low truster expects to be paid off only in the region LZL' (the expected payoff). The actual payoff for the low truster is, in this case, in the region $LZT'L''$, under the graph of the optimal truster. For the high truster, the expected payoff is in the region HZH'. However, the actual payoff is in the optimal truster's payoff region TZT', which is the real objective payoff in this example population, *less* the actual loss represented by HTH''. Thus, the average payoffs will always be greatest for the optimal truster and least for the low truster, with the high truster's average gain in between.

The high truster, because of his optimistic expectations, will enter into more interactions compared to the low truster. These interactions give the optimistic truster feedback which he can use to revise his judgements of the actual, or optimal, distribution of trustworthiness and thus allow him to move closer to the graph of the optimal truster, reducing loss, and increasing gain. This is not true for the low truster as he does not give himself the opportunity to discover the 'true' distribution of trustworthiness. It is also possible that the low truster's judgements are reinforced with each experience since it is probable that his expectations will always be met, when he does grant trust.

2.8 Deciding to Trust (or Distrust)

Apart from dispositional trust, we know that we trust others differently depending on who the (potential) trustee is as well as the situation involved, as shown in the relation below.

A trusts B about X

We also know from §2.5 that the perceived trustworthiness of an agent is insufficient when deciding on the appropriate trusting behaviour because the decision to trust involves other surrounding conditions and incentives. In this section, we will uncover how the different entities in the trust relation above affect our decision to trust or distrust.

We have discussed the dynamics of trust in §2.7. What we are concerned with in this section is the decision of whether to trust a potential trustee, or to distrust him, depending on the specific situations trusters find themselves in. This involves making the following types of evaluations:

1. Evaluating the situation.
2. Evaluating the trustee.
3. Evaluating the environment, or system.

2.8.1 Evaluating the Situation

As Luhmann said, trust depends on specific instances of the surrounding environment and the situation the truster is in, on the presupposition that "one is conscious of the possible behaviour of others as a specific problem" [Luh88] . The costs and benefits of each interaction must be evaluated within the frame of the situation one finds oneself in [Deu58].

One important factor in determining whether a positive outcome will have a higher probability of occurring is familiarity with the situation [Bar83, Luh88], e.g. the truster's past experiences in similar situations. The earlier example of the truster's familiarity with motorists in the two towns is one such instance. Furthermore, the truster's decision may also be influenced by the reported experiences of others, especially people 'close' to him.

Barber says that some of the most important reasons for perceiving that the outcome will be positive rather than negative include:

> *one's own past experiences in similar situations; the past experiences of others; the opinions held by others whom one respects; one's personal assumptions about the benevolence-malevolence of the reality one is in; and one's confidence about being able to influence the occurrence of [the positive outcome] or the nonoccurrence of [the negative outcome] through one's own actions or through available help.*

Time also affects the focus on the risks in the situation, as in another of Barber's trust hypotheses:

Barber's Trust Hypothesis 2 The more remote in time the possible occurrence of Va^- as compared with that of Va^+, the more likely it is that a trusting-choice will be made.

As explained previously, Va^+ signifies the positive outcome and Va^- the negative. Barber also mentions that there has been considerable psychological research which supports this hypothesis, i.e. that the immediate time and environment has much more influence on a person's behaviour than the future. However, it is unclear how true this is for more 'formal' decisions, such as those made by businesses, which take into account potential events and outcomes far into the future, sometimes reaching years into the future.

Evaluation of the situation is largely a case of weighing the risks and benefits of the situation and the trusting choice will be made in favour of that outcome which is most likely to occur, and one which is most likely to occur in the near future. Each situation has its own unique character and thus how we evaluate the risks and benefits for each situation is a highly subjective practice.

One can refer to the game theory literature, discussed further in §2.10, for further treatment on evaluations of strategies for specific situations. Briefly, central to game theory are the concepts of payoffs - what the truster stands to gain - and strategies - what actions are available to the truster. Each strategy has an associated payoff and the goal of a 'rational' agent is to choose the strategy that maximises his payoff. he will also need to take into account whether the interaction is 'on-off' or will the relationship be ongoing for a longer period of time. The importance of this is that there may be a higher probability of 'defection' in one-off interactions than repeated ones. We discuss this further in §2.10.

2.8.2 Evaluating the Trustee

Knowledge plays an important part when considering the potential trustee in a specific situation. The more we know about the trustee, the better we are able to determine his behaviour. For example, if Bob's Couriers had failed to deliver Alice's important documents a number of times, Alice may have

less trust in Bob's ability to deliver another vital document in future. Alice's experiences with Bob, in this case, becomes knowledge which Alice uses to make trust decisions in the future. According to Luhmann, this familiarity with situations, rooted in past experiences, makes trust possible as trust needs history as a reliable background. "One cannot confer trust without this essential basis and without all previous experiences" [Luh79].

Thus, in addition to impersonal situational considerations discussed in the previous section, we will also need to consider personal properties about the potential trustee, in terms of history, i.e. his reputation for trustworthiness. The truster will thus require information about the prospect's history relevant to trust, which, according to Hardin, are of two kinds [Har93]. The first involves what the truster knows about the past of the potential trustee and the other, information that can help formulate predictions about the prospect's future behaviour. Hardin calls the first (reputational) kind of knowledge the 'inductive' kind and the second 'theoretical'.

Hardin's first category of information, reputational knowledge, is captured in terms of generalised opinions based on opinions about the prospect's trustworthiness as conveyed by those the prospect interacted with in the past. This can also be viewed as the collective opinion about the prospect of a specific community of agents. This community may exist as a result of structural or emergent formation, based upon the members' shared context of interaction. 'Staff of the University of London' is an example of the former and 'buyers on eBay' an example of the latter.

Hardin's second kind of information is the truster's interpretations about the prospect's incentives and motivations. This kind of information may be formulated based on direct observations of the prospect's past behaviour or through word of mouth. This allows the truster to reason about how the prospect may behave given certain factors that may affect the prospect's willingness to cooperate or defect in the potential interaction.

Hardin argues that the two kinds of information above are missing from rational accounts of trust, i.e. those which say that given the same payoffs or incentives, we would all grant the same degree of trust. In other words, in rational accounts, the same situation and incentive will repeatedly provoke the same (un)trusting action. Past knowledge frames our capacity for trust, which forms the epistemological basis for reasoning, and is 'pragmatically rational', with respect to trust [Har93]. Omitting historical influence from rational accounts will, according to Hardin, cause problems of the following:

> *A problem with the encapsulated interest account of trust is that it is inherently subjective in the following sense. What is sensible for a given individual to expect depends heavily on what that individual knows, both about the past and the future of the person or other party to be trusted. Obviously, assessment of the future matter for an expectation analysis. But why the past? Partly for reputational reasons. But also because the past reveals others' capacity to trust and be trustworthy. To a very large extent, this is merely the capacity to judge the likely risks and benefits of entering trust relationships*[6].

Reputational information can assist trusters to decide whether the trustee should be considered trustworthy or not. As in the quote above, the past, forming the trustee's reputational information, "re-

[6]Lengthy emphases were removed from this quote.

veals [his] capacity to ... be trustworthy". Gambetta argues that a perception[7] of another's trustworthiness is essential to trust [Gam88a]. However, knowing whether someone is trustworthy, or trustworthy enough, is not sufficient for deciding whether to trust or not as other situational factors are involved, e.g. the amount and probability of loss for a given action. This connects back to our earlier discussion of the difference between trust and trustworthiness in §2.5. Furthermore, reputational information reveals the past of the potential trustee, which may or may not have involved elements central to the current situation. The truster will have to rely on the trustee's history and anticipate an approximated coherent behaviour.

Much of the reputational information will be based on observations of the trustee's behaviour, either directly by the trustee or indirectly via reports from other sources (whose credibility may be questionable and themselves a subject of reputational scrutiny). This means that reputational information lacks an essential ingredient which will make prediction a lot more potent – motivations behind the trustee's actions – because the external observer is not privy to the actor's thoughts. Without an insight into what motivated Bob's past actions, Alice's inductive prediction has no basis. In any case, there are means to reduce the uncertainty of outcomes by introducing external influences of motivations. The promise of punishment in case of betrayal, for example, can lessen the likelihood of potential unsatisfactory behaviour by the trustee and make predictions more acute (see §2.8.3). Hardin compares the relative strength of inductive and theoretical knowledge thus:

> *A general problem with inductive knowledge, if it is completely atheoretical, is that it cannot justify a claim that what has always happened before must happen again. Most of us are willing to live with inferences that various things will continue to happen in the ways they always have happened so far. But we are apt to suppose that there are reasons for this, merely that we do not know the reasons. The economists' theoretical knowledge about economic productivity gave an explanation (perhaps wrong) of why the trend of loyalty to Communism must eventually end. A relevant answer to the economists would have to be grounded in an alternative theoretical claim. The anti-Communists generally proposed no alternative theory, they merely asserted the certainty of the Communists' continuing irrationality.*

Both knowledge of the inductive and theoretical kind play their parts in trust decisions. The latter, the theoretical kind, to be used to strengthen or weaken suspicions or predictions based on induction. Together with the ability to understand the opposing perspective and others' motives, "trust becomes fully strategic. It is no longer merely induction on senseless facts" [Har93].

According to Deutsch, motives and intentions are prime factors to agents deciding to trust or distrust [Deu58]. Deutsch carried out experiments where trust is measured as willingness to enter into cooperation between two parties. It was discovered that the harder it is to learn and appraise the motives of the other party, the lower the level of cooperation. Thus, the truster's decision relies heavily upon evidence of the potential trustee's orientation in a particular situation [BH91].

In addition to determining trustworthiness, there is sometimes a need for the truster to know whether the trustee thinks that the truster is trustworthy [Gam88a]. This problem arises in situations involving

[7] Gambetta says that it is essential that we *know* whether the other agent is trustworthy or not. Since we assume that what is 'real' is subjective, we reflect this by saying that an agent *perceives* other agents' trustworthiness.

parties wishing to cooperate but who are ignorant of each other's motives. A simple example of this can be illustrated in the well-known Prisoner's Dilemma, described as follows.

Alice and Bob are crime suspects being interrogated in separate rooms by the police. If both confess, they each get six years in jail and if they both keep silent they each get a year in jail. If one confesses and the other keeps silent then the former, the confessor, gets off scot free while the silent one gets ten years. Alice and Bob faces the dilemma of confessing or keeping silent, based on what each thinks the other will do. The years in jail are the 'payoffs' and are shown in Table 2.4 below (the higher the number the better the payoff; 0 years is better than -1 year, that is 1 year in prison).

	Silence	Confession
Silence	(-1, -1)	(-10, 0)
Confession	(0, -10)	(-6, -6)

Table 2.4: Example payoffs in a Prisoner's Dilemma game.

The choice that will give the best payoff for both Alice and Bob is mutual silence, giving both of them a minimum sentence of a year in prison each. However, there is a risk that Bob may defect, leaving Alice as the 'sucker' with ten years in jail, while Bob gets to go home to a warm bath and comfortable bed. Here Alice needs to know whether she can trust Bob. But that is not all; Alice, who is willing to cooperate, suddenly confronts the horror that Bob may think *she* is not to be trusted, even if he is seeking cooperation! If Bob doubts Alice, then he may think she will defect and confess, and chooses to confess himself for fear of being the sucker. This will result in the tragic conclusion that both confess and spend six years in jail when in fact both are highly motivated to keep silent.

This example illustrates two points. Firstly, an apparent lack of cooperation does not always indicate a lack of motivation to cooperate [Gam88a]. In the example above, if both had confessed, neither should doubt that one had bad intentions against to other; the outcome may have come from a lack of trust in the other's motivations, perhaps due to lack of experience of each other, but there are still chances for cooperation to look forward to in their next six years behind bars. Alice's and Bob's actions during interrogation depend on their beliefs about the other and from McKnight and Chervany's trust constructs in Figure 2.2 we see the same relationship – trust behaviour depends on trust belief.

Alice's and Bob's uncertainty about each other's motives may be revealed to each other if they had communicated. This is the second point of the illustration. If they had firmly communicated their silent pact before being caught, trust in each other's beliefs would be greater, provided that they trust each other's benevolence to one other in the first place. Even if this had not been forthcoming before their capture, communicating their intentions after their mutual confession could also help cultivate a healthy relationship and the result of their communication of motives and beliefs could be used as a basis for future trust decisions. Thus communication is key in building trust [Gam88a, Mis96], as in another of Barber's trust hypotheses [Bar83]:

Barber's Trust Hypothesis 18 The communication of any of the basic elements of a cooperative system of interaction (i.e., expectation, intention, reaction to violation, or method of absolution) will tend to increase the trust of the communicatee and the trustworthiness of the communicator. The increase will be greater as the number of basic elements that are incorporated into the communication increase.

In the same volume, Barber discussed another interesting situation, which can give rise to trust between two parties; that is, the presence of mutual sentiments.

Barber's Trust Hypothesis 19 If two people have similar sentiments (like or dislike) toward a third party and each is aware that this is so, they will tend to develop positive sentiments toward one another (provided that their relationship to the third party is not an exclusive one).

For example, prior to their capture, Alice and Bob may have met each other at a party and, after a bout of hot gossiping, learned that they both had the same reasons for hating the host of the party – they both share a sentiment about the same thing. As they discover similar likes and dislikes, their trust for each other grows. This scenario becomes quite interesting when we apply this twist to Trusted Third Party protocols in secure communications – mutual trust can actually be built by having a mutually *untrusted* Third Party! This notion perhaps carries with it many assumptions that may not be realistic in certain cases, like the amount of information necessary to confirm that the sentiments shared are indeed genuine. However, it shows that Trusted Third Parties are not necessary in all situations where trust decisions are to be made. This also supports Luhmann's position that familiarity, manifested here as shared sentiments, plays a key role in building trust. As he says in [Luh79], familiarity "makes it possible to entertain relatively reliable expectations and, in consequence, to contain the remaining elements of risk as well".

In sociological research, experiments such as those carried out by Rotter [Rot67] have shown that people who trust others more are also more trustworthy. Interestingly, Rotter's experiment showed that trustworthiness, although significantly related to trust, is not the only factor that motivates trust. An individual's popularity, for example, may also invoke trust. "Trust as measured sociometrically was negatively related to dependency, not significantly related to gullibility, and positively related to humour, friendship, popularity, and especially trustworthiness".

2.8.3 Evaluating the System

As described in §2.2 and §2.3, trust can exist based on the system that constrains the situation one is in. This is called 'system trust'. It may be possible that in a particular situation, a trusting decision is made, not because there is an assumption that the trustee is trustworthy or that the situation has low or negligible risk but, rather, based on the truster's trust in the system. In other words, we may have confidence in some 'external force' to the extent that we are willing to extend trust in situations within the force's governance. An example of this is money. Underlying the concept of money is a complex economic mechanism that thrives on trust – the trust by individuals that "when he holds the money symbol in his hand, he also really possesses the possibilities that it promises, so that he can confidently put off his decision about the final use of the money and can enjoy or make best use of the complexity of the possibilities it represents in abstract form" [Luh88]. Such opportunities are made possible, not because other people are trustworthy, but because there is an assumption that a working monetary system exists.

Yet on what basis do we hold system trust? How can we trust an abstract concept such as a 'working system'? We can, because our trust implicitly depends on the proper functioning of the system's internal controls. The controls for the engine of a car are the people and machines involved in its design, production and maintenance, for example. As systems increase in complexity, there

will be increasing demand for expert knowledge to determine whether the system is trustworthy, by ensuring that proper controls are in place and working properly. Non-specialists will have to trust the economists as a reliable control mechanism to ensure the system is functioning adequately [Luh88], and the degree of consistency and predictability of the control mechanisms affects the degree of trust we have in them.

As we noted earlier in §2.2, system trust may be based on two kinds of impersonal structures. The first, structural assurances, rely on proper government, through mechanisms like regulations and contract. Some form of punishment for betrayal is necessary to maintain this kind of system trust. As Dasgupta puts it, we may trust someone because we believe he will be punished if he defects [Das88]. This, of course, relies on the credibility of the punishment mechanism and the truster's trust in the punishment agency. This 'punishment agency' may also be society at large. For example, being socially shamed may constrain the behaviour of the trustee to a certain extent. This is a form of reputation-based punishment. Since trust relationships are fragile, a threat to an agent's reputation can be an effective inducer of trustworthy behaviour. The agency may also be the party that was betrayed, or the 'injured party', to use Dasgupta's term. Punishment, in this case, may be inflicted by ceasing to interact favourably with the betrayer, provided that the betrayer wants very much to continue future interactions with the betrayed.

In determining the structures that enhance trustworthiness, Hardin noted that institutions, where system trust is placed, work well for economic rather than non-economic relations. The reason for this is that economics is far easier to assess. "There is great trustworthiness in contracts because performance is easy to assess and enforcement is relatively easy; there is far less trustworthiness in marriage in many societies and times, because performance is too hard to measure to make enforcement work" [Har96].

The second impersonal structure, structural normality, is an affective construct. It represents the internal security of the truster, that everything around him is 'normal'. Furthermore, positive feedback that he gets from around him can strengthen his non-cognitive security about the smooth-running of the system.

Evaluation of the system for system trust requires that the truster has justified confidence in the 'punishment system' as well as an internal security about the environment surrounding the situation. Other factors, too, may be included in the equation when considering the likelihood of the truster reneging on the trust. For example, in the case of a possibility of a law suit, there is the tediousness of the whole legal process, the process of finding a reputable legal counsel, not to mention the sometimes painfully heavy legal fees.

2.8.4 Trusting Trust

In the title of the concluding chapter of his book [Gam88b], Gambetta asks, "Can we trust trust?" [Gam88a]. The question refers to whether we should be optimistic when deciding to grant trust, as optimism towards others' intentions can reveal hidden cooperative motivations; and "distrust distrust", as the seemingly distrusting behaviour by others may be motivated by their uncertainty in our own intentions to cooperate [Gam88a, Luh88, Bar83].

A trusting decision postpones monitoring of the actions of others, but justification for that trust

will eventually be sought. Without the ability to gather feedback from our trusting actions, we will not have the benefit of learning from experience and be able to revise our trusting dispositions and reputational information about others. It will also make it more difficult for us to learn about when and where it is appropriate to trust, and by how much. Strangely, it is trust itself which gives us the ability to obtain this much required evidence [Gam88a]. By bestowing trust, the truster is positioning himself for reception of future evidence of others' trustworthy, or untrustworthy, behaviour. A stance of distrust will only close all doors to any communication or feedback and any chance of future cooperation, as any future evidence of trustworthiness, will not reach the distruster.

It is for this reason that some sociologists suggest the approach of 'as-if' trust, i.e. acting *as if* we trust the trustee, when we are uncertain of his trustworthiness. This regime automatically opens up channels for communication for evidence of the trustee's actions, and thus gives the truster an opportunity to discover new trustworthy partners for future interaction. Note that this attitude corresponds to McKnight and Chervany's Type B dispositional trust, as described in §2.2.

Unfortunately, no further discussions about strategies and motivations governing as-if trust have been proposed in the social sciences. Unless we are also including altruistic behaviour on the part of the truster, we believe that considerations for granting as-if trust will include the potential gains and losses in the interaction. Complete ignorance of the prospect's trustworthiness may lead to no trust being granted if the probability, or amount, of loss is high, especially if the interaction is one-off and mechanisms for sanctions are weak. However, if there exists some prior experience in with the prospect in other situations, this can be used as information to formulate opinion about the prospect's trustworthiness in a new situation. If, then, the potential gains are high and there is likely to be repeated interactions, then as-if trust may be a beneficial strategy for the truster.

Having said that, it is easier for untrustworthiness to be revealed than trustworthiness. As Gambetta said, while "it is never that difficult to find evidence of untrustworthy behaviour, it is virtually impossible to prove its positive mirror image". Furthermore, trust is "a peculiar belief predicated not on evidence but on the lack of *contrary* evidence – a feature that ... makes it vulnerable to deliberate destruction" [Gam88a]. Thus, acting optimistically may, in addition to opening ourselves to evidence of trustworthiness, also put us in a situation of vulnerability, especially when the risk of trusting is high. On the other hand, distrust can prove to be equally damaging. Gambetta explains [Gam88a]:

> *Once distrust has set in it soon becomes impossible to know if it was ever in fact justified, for it has the capacity to be* self-fulfilling, *to generate a reality consistent with itself. It then becomes individually 'rational' to behave accordingly, even for those previously prepared to act on more optimistic expectations. Only accident or a third party may set up the right kind of 'experiment' to prove distrust unfounded (and even so ... cognitive inertia may prevent people from changing their beliefs).*

Thus, if the risk of trusting in the now can be tolerated for a more lucrative future benefit, then acting as if one trusted can be a reasonable course of action in situations of uncertainty. Trust uncovers the trustworthy and opens doors for profitable avenues, while doubt "is far more insidious than certainty, and distrust may become the source of its own evidence" [Gam88a]. We find this also in Barber's seventeenth trust hypothesis [Bar83]:

Barber's Trust Hypothesis 17 The correctibility of an individual's perceptions of the situation will be greater when the individual begins with the perception that the situation is one of mutual trust (and it is not) than when he begins with the perception that the situation is one of mutual suspicion (and it is not).

2.8.5 Miscellaneous Motivations for Trust

Even if one could calculate, with a reasonable degree of accuracy, the risks and benefits involved in any given situation, the motives and incentives of the parties concerned and the effectiveness of systemic infrastructures that are in place, there may be reasons for trusting that defeat an external observer's comprehension. There is always a chance that the truster based his trust on hidden subjective and affective factors.

Barber gave a list of circumstances that might motivate a person's decision to trust, to take a choice that "might have potential negative consequences that would outweigh the potential positive consequences" [Bar83]. Some of them are listed below.

Trust as despair Or choosing from the lesser of two (or more) evils.

Trust as social conformity Trust is expected in certain social situations.

Trust as innocence Perhaps there was a lack of appreciation of the risks involved.

Trust as impulsiveness Acting "in terms of here and now" [Bar83] instead of pondering on the future outcome.

Trust as virtue When to trust is looked upon as a virtue.

Trust as masochism A preference of pain over pleasure, perhaps.

Trust as faith Faith in God's help, or that whatever turns out is fated. Barber calls this form of trust "extreme".

Thus, a trusting behaviour need not necessarily be observed as an act made after a risk-benefit evaluation. A display of trust may have instead been based on something more emotionally significant or on pathological reasons.

2.9 Rationality and Trust

The previous section shows that sometimes the decision to trust may or may not be based on 'rational' motivations. It is important to know whether trust is always motivated by rational motivations so that our trust model can be designed accordingly. If we assume that trusters are not always rational when making trust decisions then our trust algorithms must allow for 'rational' choices to be overridden when the truster decides irrationally. It also impacts our evaluation of others when making judgements on trustworthiness - can the other person be expected to behave similarly given the same motivations and incentives?

Definition 7 (Rational choice) *A rational choice is made when the chosen action is one that will produce the highest 'utility' for the actor [Mis96]*

Turner gives the following as assumptions in rational choice theory [Tur02]:

1. Humans are purposive and goal oriented.
2. Humans have sets of hierarchically ordered preferences, or utilities.
3. In choosing lines of behaviour, humans make rational calculations with respect to:
   - the utility of alternative lines of conduct with reference to the preference hierarchy.
   - the costs of each alternative in terms of utilities foregone.
   - the best way to maximise utility.
4. Emergent social phenomena – social structures, collective decisions, and collective behaviour – are ultimately the result of rational choices made by utility-maximising individuals.
5. Emergent social phenomena that arise from rational choices constitute a set of parameters for subsequent rational choices of individuals in the sense that they determine:
   - the distribution of resources among individuals.
   - the distribution of opportunities for various lines of behaviour.
   - the distribution and nature of norms and obligations in a situation.

In short, according to rational choice theory, one chooses the 'maximised' option based on the presumed utility of the action and which is contingent upon the actions of others. We can also think of the latter in terms of risk. Thus rational choice involves risk and benefit. This is relevant to trust as trust is "a purposive behaviour aiming at the maximization of utility under risk" [Mis96]. Furthermore, if we are able to assume that a potential truster is 'rational', then rational choice theory may be able to help improve the truster's prediction about the prospective trustee's behaviour and consequently make the appropriate trusting decision. However, this may not be always a reasonable assumption, i.e. the trust prospect may not behave in a rational manner. We investigate this further in the game theory section in §2.10.

We now ask under what conditions can rational choice theory work for trust? As mentioned above, rational choice is only concerned with the goals of the truster and the risks he faces in trusting the trustee. The choice he makes is also based on what he assumes the trustee will do. The question is, thus, whether it is rational to trust or distrust for a given situation.

Using the simple Prisoner's Dilemma game in Table 2.4, one might conclude that the rational choice would be to confess, to minimise the jail sentence in case the other person also confesses. This is acceptable if the prisoner holds a Hobbesian view of the world; that is, that the state of nature is a state of war [Hob47] and morality cannot sustain cooperation. It is then "rational to bet on self-interest when one cannot predict what one's fellow will do" [Hel67]. However, if one shares the view of Locke, that we are obliged to act within the confines of Nature and not force that state into a state of war, then rationality dictates that we keep silent [Hel67]. In short, Hobbes would deem it rational to distrust, while for Locke it is our moral obligation, and thus rational, to trust.

In economics, trust is regarded as an effective lubricant for exchange. As Dasgupta says, "trust is central to all transactions" [Das88]. Trust is also viewed sometimes as a replacement for formal contracts or 'implicit contracting' [Arr74]. Trust is further regarded as a form of social capital since it reduces the cost of monitoring and sanctioning [Mis96]. Thus, in economic exchanges, it is perhaps rational to trust in situations of dilemma if mutual benefit is to be gained.

Although one can analyse the costs and utilities of actions in a given situation, rational choice theory lacks the depth to determine one's trusting choice as trust is ingrained deep in the actors' subjective view of the world or, more generally, the beliefs that he holds. Furthermore, as we discussed in §2.8.2, rational accounts of trust usually omit the ability of the trustee to obtain relevant knowledge, e.g. reputational information, to make trust judgements about the potential trustee [Har93].

One well known method used to study rational choice is the study of games, or generally known as game theory [MvN47, Gin00, Mye97]. We discuss the role of trust in game theory next.

2.10 Game Theory

In game theory, one starts by specifying 'games', or situations within which two or more agents are given possible 'moves' or strategies and incentives of varying degrees for each strategy. For each strategy is a 'payoff', i.e. the 'utility' (e.g., an amount of cash) that an agent gains, given what the other players' moves are.

The game theory literature is vast, covering many variations of games and new ones are continually being discovered. Due to the scale of the subject, we will not cover it in depth here. What will be of interest to us is the how game theory can be used to inform us on the dynamics of trust and what the limits are in its ability in doing so.

A repeated game is a game where the players expect to meet again in future within similar circumstances whereas a one-shot game is where this expectation doesn't exist. An example of a repeated game is business between the local grocery store owner and his local neighbourhood where he expects repeat business from his customers. In this example, the incentive of repeat business may motivate the owner to maintain an acceptable level of customer service and product quality. A customer who beliefs that the owner reasons in this manner may then have more trust in the store's service and quality – the customer's belief in the store owner's motivation influenced her trust in the store. Compare this to a one-shot game where the store owner does not expect the customer to return, and may, as a result, try to maximise his profit for each one-shot transaction by selling cheap, low-quality goods at a high price.

To illustrate our discussion concretely, we will use the famous Prisoner's Dilemma (PD) game. We describe the game then follow that with a study of its trust properties.

In the PD game, Alice and Bob are partners in an armed robbery and they have just been arrested for it. Luckily, the police do not have any evidence against either of them on the robbery. They do, however, have evidence against them for the theft of the getaway car. Using this as a bargaining tool, the police makes the following offer to Alice. Confess to the robbery, implicating Bob, and Alice gets away scot free, provided Bob remains silent and doesn't confess. If both confess then they both get five years in jail. If both of them stay silent then both of them get two years for the car theft. The

same offer is made to Bob.

The potential jail terms can be expressed in terms of 'utility' values, where the lower the jail term, the higher the utility value to Alice and Bob, for example:

Jail term	Utility value
0 years	3
2 years	2
5 years	1
10 years	0

The PD can now be modeled as shown in Table 2.5, with the utility values used as 'payoffs'. In each box, the first value in the payoff pair represents the payoff to Alice should she choose the strategy shown by the row it is in, and the second value the payoff to Bob for the strategy column the value is in.

	Bob confess	Bob silent
Alice confess	(1,1)	(3,0)
Alice silent	(0,3)	(2,2)

Table 2.5: Example Prisoner's Dilemma game matrix.

We first look at the one-shot (non repeated) PD game. If the assumption is that the players in a PD are rational, then both Alice and Bob will seek to maximise their utility values in their moves. We assume here that they are both not able to communicate and must make simultaneous moves. If Bob confesses then Alice's possible payoffs are 1 if she too confesses, or 0 if she stays silent. Thus, choosing confess will give her the best utility value in this instance. If Bob chooses to stay silent then Alice's possible payoffs are 3 if she confesses, or 2 otherwise. Again, her best move is to confess. In this PD example, Alice is better off confessing regardless of Bob's actions. In game theory, her strategy to confess is called the game's dominant strategy, because regardless of what Bob chooses, confession dominates over keeping silent as her best strategy. The best move (or moves) given what the other player does is called the game's equilibrium.

Since we are assuming that all agents are rational, we must also assume that Bob will come to the same conclusion, i.e. that his best move is to confess. Following these strategies through will result in the both of them receiving 5 years each. On the positive side, this means that neither were enough of a 'sucker' to think the other is going to stay silent. On the negative side, they both missed out on the higher payoff of 2 years each had they both remained silent. However, because we are assuming purely rational agents that seek to maximise their utility, the silent-silent strategy gets ignored as a viable alternative. This is the negative aspect of assuming that agents are purely rational. In reality, Alice and Bob may have anticipated being in this situation and agreed that they both should choose to stay silent to attain the best possible outcome, that is 2 years each.

Assume that Alice and Bob did agree beforehand to stay silent if they were caught. Should Alice, now caught and facing a jail term, stick to the agreement? If she does, would Bob play her for a 'sucker' and confess, getting away scot free while she gets 10 years for staying silent. This is where beliefs about trustworthiness enter the equation. Alice will most likely stick with the agreed strategy to stay silent only if she strongly believes that Bob will also do the same, and likewise for Bob.

The higher the trust between both parties, the stronger this belief and the more likely they will both cooperate.

The strength of this belief (that the other person will cooperate) is higher in repeated games. If Bob expects to rob another bank with Alice after being in the current situation of being caught, regardless of the outcome, then there is incentive for Bob to cooperate as this will strengthen trust in Alice towards him, thus increasing the chances of cooperation in future. By this reasoning, Alice may be confident that Bob will cooperate in this round of the PD game, because of her belief that Bob is expecting to work with her in future. Thus, it can be said that repeated games provides an incentive to trust.

However, repeated games may also be finitely or infinitely repeated, and this impacts the players' beliefs in a game. For example, if the PD games is played for ten moves, then the rational Alice, knowing that the tenth move will be her last, will confess on the tenth move (in other words, not cooperate) because there is not motivation for her to maintain a good relationship with Bob any further, and can maximise her payoff by confessing. In other words, confessing again becomes the dominant strategy because the game will not be repeated after the tenth move. Bob will also arrive at the same conclusion and decide that in order not to be suckered into cooperation, he will have to confess on the ninth move. Unless she is willing to be the sucker, now Alice will have to confess even earlier than that, on the eight move. This situation can be 'unraveled' until Alice and Bob reach the inevitable conclusion that they are both left with the dominant strategy of confessing on the first move anyway. This unraveling does not happen if the length of the relationship is unknown, that is, if the game is infinitely repeated.

The unraveling phenomenon above is based on assuming that both Alice and Bob are purely rational. However, the assumption of strict rationality is not always a reasonable one as it limits the possible motivations behind players' decisions during the analysis and does not explain why players may choose the seemingly 'irrational' move to cooperate (in the PD example) and increase their payoffs. As we discussed in §2.8.5, there are many reasons for why one would trust another, and the reason may not be a strictly rational one.

In the PD game above for example, Bob may decide to stay silent and hope that Alice will confess because, one might imagine, they are lovers and Bob is sacrificing himself for Alice to go free. Thus, for analysing motivations behind trust decisions, one must take into account beliefs and motivations that may not be rational.

Good provides a more in depth discussion on the psychology of trust in PD games in [Goo88]. His discussion is based on controlled experiments that were carried out in social science laboratories. In it, cooperative behaviour is taken as an indication of some level of trust in the agent that is cooperating. Good claims that although the Prisoner's Dilemma is only a narrow representation of the situations that exist in real life, it does provide certain lessons that are relevant to trust. In particular, a certain level of cooperation, and to a certain degree, trust, can develop where:

1. The long-term interests of the participants are stressed.
2. Only small initial or additional rewards are at stake.
3. There is no potential for threat and great potential for successful communication in that the

ambiguity of the situation is reduced.

4. The participants are in free and easy contact.

Thus, even limited experiments with game theory can highlight situations where cooperation can arise. However, care must be taken in drawing conclusions about trust from cooperative behaviour. As Good puts it[Goo88]:

> *Cooperative behaviour by itself is not, of course, necessarily a sign of a cooperative mentality. It could be cooperative by chance rather than design. Similarly, a lack of cooperation need not indicate an uncooperative mentality; nor need it represent some deception or breach of trust. Consequently, while cooperation and trust are intimately related in that the former is a central manifestation of the latter, the former cannot provide, for either the actor or the analyst, a simple redefinition of trust.*

Game theory, while being able to show when, under assumptions of rationality, cooperation can occur, does not capture irrational behaviour, and neither does it succeed in representing motivations beyond profits and payoff. For these reasons it is limited in scope for studying the dynamics of trust. Nevertheless, the game theory literature is vast and new situations (games) are still being discovered and analysed.

2.11 Chapter Summary

In this chapter, we have attempted to discover the meanings and mechanisms of trust as a social phenomenon. As we have seen, there are a large number of social accounts of trust that may or may not overlap with each other. Of particular interest are the wide-ranging definitions of trust. These different definitions of trust have been put together by McKnight and Chervany as a set of inter-related constructs [MC96]. Misztal gives a good summary of how trust has been perceived by social scientists:

> *What integrates all the above definitions of trust is their common emphasis on the importance of several properties of trust relationships. The main common characteristic ... is its 'dependence on something future or contingent; future anticipation'. ... they require a time lapse between one's expectations and the other's action. ... that to trust involves more than believing; in fact, to trust is to believe despite uncertainty. ... always involves an element of risk ... from our inability to monitor others' behaviours, from our inability to have a complete knowledge about other people's motivations and, generally, from the contingency of social reality. Consequently, one's behaviour is influenced by one's beliefs about the likelihood of others behaving or not behaving in a certain way rather than solely by a cognitive understanding or by firm and certain calculation.*

Although the quote above may serve as a useful guide to the concepts that form the basis of many definitions we find in the literature, it is not to be taken as an all-encompassing objective statement of fact. The reasons are, according to Cvetkovich and Löfstedt that, firstly, the definitions are made within bodies of work that have their own scopes and goals that may not be compatible with other

people's work. Secondly, some of these definitions arise from particular assumptions on the internal mechanisms of trust, which may not be equal to other people's assumptions on how trust works [CL99].

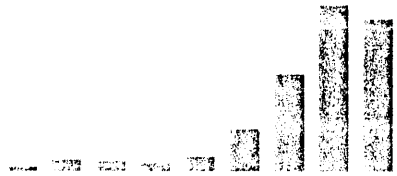
We have also delved into the possible representations of trust and the dynamics of a trust relationship, which is fragile in nature. When it comes to situational decisions to trust, several factors that impact our evaluation of the situation were looked at, including motivations for trusting that may not be seen as 'rational'.

This survey has been driven by the original questions presented at the beginning of the chapter. Therefore, there are other areas of trust that may have been omitted, such as the issue of trust and morality and trust between intimate partners. Although interesting topics in their own right, we have skipped them because we feel that their impact on our trust model for distributed systems is minimal.

The workings of trust are complex and differ in many subtle ways from one situation to the next, from one party to the next. There is a real need to look at each situation more closely to identify how trust functions specifically for those situations. To achieve this, one must rely on learning through experience in those specific situations.

This chapter provides lessons on how trust models can and should be developed for distributed computer systems. The definitions of trust presented can be a useful guideline to applications that work with the concept of trust, such as applications that use public-key certificates. In fully automated distributed applications, such as the emerging agent technology, our discussion on how trust decisions are made is important as agents will have to make these decisions for themselves. Discussions about the subjective areas of trust are also important in cases where there humans are involved – we need to know which parts of a trust model cannot be objectively represented and which need manual feedback. In Chapter 4, we will compare existing trust models to our findings in this chapter.

Chapter 3



Reputation

He who stands by what he has allowed to be known about himself, whether consciously or unconsciously, is worthy of trust.

Niklas Luhmann [Luh79]

To make effective and informed trust decisions, each individual will need to rely on sources of information other than themselves. In society, mechanisms exist to assist us in making these trust decisions. One of the most important of those mechanisms, as mentioned in the last chapter, is reputation.

Reputation is a *social control* mechanism. In Rasmusson's words, "in a socially controlled system, it is the participants themselves, the agents, who are responsible for cooperatively maintaining security" [Ras96]. Therefore, reputation, as a social control mechanism, is not only an approach to distributed security enforcement, but also, as we have discussed in the last chapter, an important class of information for trust decisions.

In commerce and economics, where reputation is commonly encapsulated in brand names or names of companies [Tad99], it has been described as a form of capital [Das88]. Names that carry good reputation behind them carry real value and are sometimes traded. Likewise, names that carry bad reputation are sometimes discarded and replaced because they have become 'untrusted'. Tadelis likened the reputation behind a name to assets that belong to the name's owner. It is normally referred to in economics as an 'intangible asset' and have a real monetary value associated with it. In a company's balance sheet, this would usually be called 'goodwill' [Tad99].

In this chapter, we will look a little deeper into reputation and its related constructs and how they support the process of making trust decisions. We start by dissecting the notion of reputation and its contents. We then look at how reputation is propagated through social relationships via *recommendations*. A short section on the semantics of communicated reputational information follows that. Finally, before the conclusion, we discuss the notion of names and its impact on the reputation mechanism.

3.1 What is Reputation

The following is a definition of *reputation* from The Oxford Modern English Dictionary [Swa92]:

> *1. what is generally said or believed about a person's or thing's character or standing (has a reputation for dishonesty).*

Reputation refers to the general expectation about an agent's future actions based upon his past actions. In other words, "Alice's reputation of Bob is her expectation of future interactions with Bob", based on what she knows about Bob's past [Dai].

In economic terms, this expectation can be converted to some monetary value [Tad99]. By building a good reputation for himself, thus building positive expectation in others about his own behaviour, can influence others' decisions on whether to enter into business transactions with him, for the agent with positive reputation, this means work or business for him, which translate indirectly into money. As another example, from blau [Bla89], a high credit rating, hence a good financial reputation, will benefit the reputation holder when applying for a bank loan. In this work, we will define reputation as follows:

> **Definition 8 (Reputation)** *A reputation is an expectation about an agent's behaviour based on information about or observations of his past actions.*

Reputation can be thought of as a group's or society's 'general' opinion about the subject. However, the measure of 'general-ness' or majority may be impossible to gauge in a large and dynamic group such as the Internet. Therefore, such beliefs about the generality of a reputation can only be a subjective perception. For example, Alice's perception of Bob's reputation may be the result of information obtained from sources known to her. These sources may, to Alice, be representative of the majority. However, this may not objectively be so, as Alice's belief need not depend on an objective fact, but Alice's presumption of the truth. Therefore, due to this subjective nature of a reputation, we will assume that reputational information about a specific subject may differ from one opinion holder to another. For example, Alice may believe that Bob has a reputation to be completely honest while Cathy may hold that Bob's reputation is not completely benign.

Reputational information may also be aggregated *solely* from direct experiences with the subject. If Alice's sources of information cannot provide any reputational information about Bob then her opinion of Bob can only come from direct experiences with him. Because, as we have mentioned above, reputational information is subjective and need not be globally consistent, to Alice, her own opinion is as valid as those gathered from third parties.

However, Alice, and her experiences with Bob, may be a provider of Bob-related reputational information to other people who may ask about Bob from Alice. In society, this form of exchange is carried out all the time, through word of mouth, recommendations and gossip. What makes it powerful is not whether the information is completely correct or not, but because it gives us the opportunity to minimise risks in our everyday interaction by choosing to rely (or not rely) on word of mouth based on how much we trust our sources. As Resnick and Varian comment [RV97]:

It is often necessary to make choices without sufficient personal experiences of the alternatives. In everyday life, we rely on recommendations from other people either by word of mouth, recommendation letters, movie and book reviews printed in newspapers, or general surveys

The weighting we place on the information and the context in which we use it allows us to rely on it, discard it or aggregate it with information from other sources correspondingly. In this work, we will attempt to incorporate the social mechanism of word of mouth in on-line interactions so that they can benefit from this powerful social control mechanism. We will collectively refer to the mechanisms of propagating reputational information (through word of mouth, gossip and recommendations) as the Recommendation protocol, and the messages simply as Recommendations.

Definition 9 (Recommendation) *A recommendation is communicated opinion about the trustworthiness of a third party agent.*

3.2 Stereotypes

A question arises when we encounter complete strangers without any available reputational information. How do we form a picture of the stranger during initial encounter? In the absence of reputational information, stereotyping is used. Racial and gender stereotyping is a common gauge when interacting with strangers [Mis96]. In a virtual world, the background of a person may be more difficult to ascertain. However, this may still not deter the tendency to base our judgements on stereotypes. Misztal says that the process of classification and the attachment of expectations become more complex and blurred in the advancement of globalisation. This is due to a high rate of encounters and 'living with' strangers which is a mark of 'modern living'. However, even though it is increasingly hard to label people, the tendency within society to do so is unfazed, for stereotyping "helps people to come to grips with social reality and to make it more comprehensible" [Mis96]. Thus, even if physical appearances may not be forthcoming in on-line interactions, other information may be used to base stereotypes, like domain names of e-mail addresses or writing styles (e.g. proficiency in the English language as an indicator of level of education).

The last example can be seen in the algorithms of email spam filters, one of the most popular being the Bayesian approach [AKC+00]. Emails from popular free web-based email domains such at hotmail.com rate highly as spam due to the high number of spams originating from such email accounts. The spam filter thus learns to stereotype hotmail.com (for example) as a notorious spam source and adjusts its filtering policy to reflect this.

3.3 Representation

In the Recommendation protocol, reputational information is exchanged between interacting parties. To allow us to build this protocol for online interaction, we will require an understanding of what constitutes a piece of reputational information.

From our discussion in the previous section, we can already highlight several types of information, some of which may be private information held only in local databases for decision making:

Subject Name This may be a reference to an individual person, or company (e.g. Joe Bloggs), or it can also be the uniquely identifying characteristics for supporting stereotypes, e.g. Asian male scientists represented as a set of identifying keys, such as {Ethnicity=Asian, Sex=male, Occupation=scientist}.

Expectations This represents an expectation or history about the name. For example, 'Bob is highly trustworthy', or 'Bob is dishonest'. This should be treated as a subjective belief of the source of this information as the perception of Bob's honesty and trustworthiness may differ from one agent to another.

Historical Information This may contain past activities of the subject, for example '3 convictions for drug trafficking'.

Original Source This contains the name or pseudonym of the original source of this information. This is essential for determining the authenticity and reliability of the information by looking for reputational information about this particular source.

Secondary Sources As information is propagated, there is a chance that the information gets corrupted. Chinese Whispers is an example of this problem. Therefore, names, or pseudonyms, of the intermediaries that carried the information can help determine whether the information is reliable, by either looking at the reputations of the intermediary sources, or deciding on the length of the propagation chain.

Timestamp Old information may be unreliable and outdated. Therefore, the creation time of the information is essential.

Other kinds of information have been suggested. Dai describes a reputation system as one consisting of a set of entities, each having a reputation and method which he uses to update his opinion of others [Dai]. For each reputational information are three values:

1. An *operating value* that indicates the maximum profit his current reputation can gain him in the future.
2. A *throw-away value* which is his maximum obtainable profit, including gains from actions that will potentially destroy his reputation completely.
3. The *replacement cost* of recreating an equivalent reputation if it was destroyed.

Dai's model is biased towards economic transactions, which allowed him to suggest that Alice's reputation of Bob can be represented by a graph where a point (x, y) represents the expected utility y from paying Bob x amount of money.

3.4 Forming an Opinion of Others

According to Misztal [Mis96]:

> *A person's reputation is created out of a multitude of events and actions past and present, which build up an image of a member of a community. This shared image*

of a member of a community, which is stable over time – that is, her or his reputation, affects how people behave and what they expect from a given person. The construction of a reputation within the community depends on the group members having complete information about one another's past behaviour.

Although it may be increasingly difficult to obtain complete historical information about someone, reputation is essentially based on how much we know about the person – knowledge about his past actions. Furthermore, the strength of the reputation depends on the stability of this 'image' about him over time. How we obtain this information and its reliability depend on the context of the groups and two avenues are outlined by Misztal.

The first method is through personal interactions and informal contacts. This happens in "smaller and more homogeneous" communities, where the strength of personal relationships are considered important. Hardin's 'thick-relationship' theory of trust supports this [Har93]. Thick relationships concern our personal ties with people close to us, for example with the members of our family or close friends. In these relationships, familiarity and frequent experiences allows us to judge their limits of trustworthiness well – we know with confidence who we can trust and for what. In fact, such relationships in small informal groups, like kin groups, Misztal considers reputation to be no different from trust. Hardin sees thick-relationships as playing two important roles. The first is its role as *one possible source* of knowledge about the trustworthiness of another, or a kind of shallowly transitive chain of information. For example, Alice's thick relationship with Bob is a source of information about Cathy, whom Bob knows about, that Alice believes she can rely on. The second role of thick-relationships is as "*one possible source of incentives to the trusted* to be trustworthy". This second role is based on the workings of an iterated prisoner's dilemma where mutual cooperation over multiple interactions is desired [Har93]. Thus, thick-relationships are important sources of information which is then used to form reputational information about others.

The second method of obtaining reputational information is through formal evidence of reputation. This is used when we are dealing with members of a larger community where interpersonal relations do not (yet) exist. Examples of these formal evidences include referee reports in job applications and credit information used in financial loan applications.

Any information collected is then subject to interpretations by those seeking to form an opinion about others. The perception of whether one has a good or bad reputation will then depend on a variety of standards, frameworks and contexts. Having a good reputation in one specific community will not necessarily mean that an equal perception will be held by members of a different community, even when there is communication about the subject's past behaviour. However, in some cases, the 'credit' of reputation can be transferred from one group or community to another. An example in [Mis96] cited the practice of a number of retired High Court judges in Australia, who sell their reputation to private companies by having them as hired consultants, which in turn boosts the reputation of the company itself (by having these former judges as consultants).

For strangers in a new group, however, there may not be enough or even any information available to the group members to form a reputation about the newcomer. Hence, there may be few opportunities for the stranger to demonstrate trustworthiness, especially if suspicion is prevalent amongst the group

members. In this case, it is contingent upon the newcomer to undertake initiatives to prove his worth. An example is the case of immigrants who lack reputation in a new country and at the same time arrived without any external guarantees from their country of origin. This becomes a barrier for them to enter the labour market in their new host country. As such, members of the same ethnic community tend to work in the same circle since we tend to like and can better predict the behaviour of those most like ourselves [Har93]. In tightly knit ethnic communities, local entrepreneurs give newcomers experience-building opportunities to build a reputation for themselves. This 'bootstraps' the reputation building process. Further examples of this are given in [Mis96].

For immigrant groups without local resources, the process of building a reputation is slower and more laborious. Members of these minorities will have initially to cope with stereotypes that the host community has formed about them.

3.5 Use of Reputational Information

Reputational information can either be solely for private use or shared between collaborating parties. The former may be desirable in cases where privacy of opinion is desired. For example, we may not want people to know what we think about a certain member of the community for fear that it may jeopardise our relationship with him or her. Some public-key cryptographic protocols use this method, for example PGP [AR97].

However, the power of reputational information can be harnessed when it is shared. The analogy is how people share information by recommendations propagated through word of mouth. For example, a close friend who is more knowledgeable in matters concerning automobiles may be one of our 'trusted' sources of information about automobiles. The 'power' of this mechanism comes from its ability to function as an information 'filter'; that is, the information we receive is usually perceived to be reliable because we sought the information from sources that we trust more than any other. This trust comes from existing relationships with them, or through the reputation mechanism itself, i.e. that the sources have a reputation to be 'trustworthy' or 'reliable'. The former was discussed earlier as Hardin's 'thick relationships'. Some early experiments with automating the recommendation or word of mouth mechanism have been carried out within an area known as *collaborative filtering*. These systems are also referred to as 'recommender systems'.

3.6 Collaborative Filtering

Collaborative filtering [RV97, SM95, Sha94] attempts to augment the social mechanism of word of mouth. It assumes that people's tastes, in music or food, for example, are not randomly distributed and tries to provide recommendations to individuals based on other people with similar tastes. The basic working of a recommender systems is as follows. A new user provides a number of recommendations to the system and these are stored in a database. When a user asks for a recommendation, the recommender system looks through the user's recommendations and then matches them with other users' recommendations. Several algorithms have been proposed for matching and predicting user tastes [Sha94]. The accuracy of the system depends on the number of recommendations it receives in total and the number of recommendations a requesting user has submitted.

The advantage of this system is that instead of asking several friends whom we know have similar tastes to us, recommendations can be made based on similarity measures that are, in turn, based on a larger number of profiles. However, privacy may be at risk, especially if recommendations are related to information that users may consider to be confidential or sensitive. This is due to the centralised nature of the system, which maintains profiles of a large number of users.

Collaborative filtering systems have been designed to handle the large amount of information that users may come across and tries to minimise the problem of 'information overload' by providing a social filtering mechanism. Thus, its focus is on the content itself, and less on its source. Herein lies its difference to reputation-based systems.

Collaborative filtering and reputation-based systems can both be extremely useful in different contexts; the usefulness of the former arises when the emphasis is on the content, and the latter can be used when the source of information is a more important factor. They are thus complimentary social mechanisms for use in global open distributed systems. Indeed, researchers have already began looking into enhancing collaborative filtering systems with reputational information [MA04, MB04].

Since our focus is the reputation for trustworthiness, reputation-based systems are better suited for our purposes. As we discovered in the last chapter, trust is a subjective belief and there may be a myriad of reasons for making trust decisions. This means that the control over trust decisions should be with the agent itself, and not recommended by some central database system. One important handicap of collaborative filtering systems with respect to this problem is its lack of historical information that forms the foundation of reputational information. Furthermore, a centralised approach assumes that trust decisions can be made using a generalised algorithm for arbitrary agents. An agent may also be reluctant to submit recommendations about whom he trusts to a central database as such information is usually considered sensitive – a trusted companion would be a more acceptable recipient of this class of private information. Therefore, for trust decisions, a distributed reputation system is a more suitable approach.

3.7 Semantics of Recommendations

One problem with recommendations that one party may pass to another is how the recipient interprets the recommendation. For example, how should Bob interpret Alice's recommendation of 'Alice trusts Cathy'? One interpretation could be that Alice has actively trusted Cathy in the past, i.e. has taken a course of action that precluded a certain level of trust in Cathy by Alice. The other interpretation could be that Alice believes Cathy is trustworthy. Is there a difference between the two interpretations? The answer is yes. The first interpretation describes an active trust action by Alice, the second describes Alice's belief. We will refer to the first as an 'active' statement and the second as a 'belief' statement.

Both active and belief statements can be valuable information. However, the active statement merely reports that Alice has carried out trusting actions with respect to Cathy without revealing any motives behind those actions. This can open the statement up to further subjective interpretations, based on assumptions about Alice and Cathy, and the possible situation they were in. A belief statement may be less ambiguous in this respect as, when further qualified, it can convey a general opinion Alice has about Cathy in a given situation. This 'general opinion' form allows reputation, as defined earlier in

this chapter, to be communicated.

A preference for using one term instead of the other may affect the meaning or semantics strength of the statements we make about trust.

3.8 Names

The ability to identify an individual agent uniquely is vital in any reputation system. Without it, reputational information is meaningless as we would not know for certain about whom the information is about [Kha99]. A common identifier is the IP address, but this has several shortcomings: 1) IP addresses can be spoofed easily [CER96]; 2) some IP addresses are temporary [MGM03]; 3) IP addresses are bound to the host, not the user, which means any reputational information is not transferred when the user moves from one host to another.

An alternative method of naming is to use pseudonyms. A pseudonym (or 'nym') allows an agent to maintain an identity but still be able to hide his real world names, i.e. to remain anonymous. A public key is an example of a pseudonym, as in Chaum's words [Cha81]:

> *A digital* pseudonym *is a public key used to verify signatures made by the anonymous holder of the corresponding private key.*

Pseudonyms are the primary identifiers used in anonymous remailers, i.e. servers that receive and forward a message through intermediaries before sending it to the intended recipient [Cha81]. Remailers implement well known protocols such as Onion Routing [GRS96] and Chaum's mix networks [Cha81, mix]

In reputation systems, 'cheap' pseudonyms are a problem [FR01]. It is easy for a user to change identities on the Internet. For example, obtaining a new email address from a public free email provider is a trivial exercise that any user can carry out. This allows users with bad reputation (that is reputation attached to his pseudonym) to stop using his pseudonym and create a new one, giving himself a blank slate to work from. This allows malicious agents to return into a community after mounting an attack within it. Various solutions to this problem have been formulated. For example, in the eBay auction community, certain services to sellers require a track record to be eligible. Furthermore, a rating system allows buyers to identify new sellers and take precautions if necessary.

IP addresses and pseudonyms are examples of global names. While a globally unique name makes identification easy, it can cause problems when the name space is limited and where name clashes are possible (e.g. "John Smith" is owned by more than one person in this world). An alternative is to use a distributed approach to naming. A well known method is SDSI, the local names approach used the SPKI public key infrastructure [E+99].

The idea behind SDSI is that names need not be globally unique – it is sufficient that it is unique to the person using that name to reference the entity bound to that name. Through a combination of the use of globally unique public keys and chaining name spaces, local names that are meaningful to the user can be used as a globally unique name to be used by computers. The following is a Fully Qualified SDSI Name for the name 'therese' [E+99]:

```
(name (hash sha1 |TLCgPLFlGTzgUbcaYLW8kGTEnUk=|) jim therese)
```

The name 'therese' is defined in jim's namespace, whose name is in turn defined in the namespace of the entity referred to by the given public key hash string. The hash string is assumed to be globally unique. Thus to reach the locally defined name 'therese' one needs to traverse the chain 'public key' → 'jim' → 'therese'.

Issues about naming on the Internet centres around the need for anonymity. The need to remain incognito is offset by the need to obtain services. Without the ability to identify clients, servers will not be able to provide services if provision of that service relies on the client's reputation. Even at the basic level of communication, a sender of a message will not be able to send to the receiver if he does not know the recipient's ID or destination. Despite this, Internet users have demonstrated that they are willing to compromise some anonymity in return for service. File sharing platforms such as FastTrack and Gnutella [gnu] reveal each user's IP address, yet thousands of users log on to download copyrighted content, even with threat of legal action from the music industry. For example, Gummadi et al identified over 24,000 users and over 1.6 million requests in a trace of the Kazaa network spanning 203 days [GDS+03].The P2P software Kazaa even includes spy/adware, yet it is the most popular file sharing software at the time of writing.

The willingness to barter private information for service has been exploited by identity thieves. Foley et al [F+03] defines three main forms of identity theft:

- *Financial identity theft* involves the imposters use of personal identifying information to establish new credit lines in the name of the victim. This person may apply for telephone service, credit cards or loans, buy merchandise, or lease cars and apartments. Subcategories of this crime include credit and checking account fraud or takeover.
- *Criminal identity theft* occurs when a criminal gives another persons personal identifying information in place of his or her own to law enforcement. For example, Susan is stopped by the police for running a red light. She says she does not have her license with her and gives her sisters information in place of her own. This information is placed on the citation. When Susan fails to appear in court, a warrant is issued for her sister (the name on the ticket).
- *Identity cloning* is the third category in which the imposter uses the victims information to establish a new life. He or she actually lives and works as you. This crime may also involve financial and criminal identity theft as well. Types of people who may try this fraud include undocumented immigrants, wanted felons, people who do not want to be tracked (i.e. getting out of paying child support or escaping from an abusive situation), and those who wish to leave behind a poor work and financial history and start over.

A real world example of identity theft is one where spam messages claiming to originate from a bank invite users to click to visit the bank's website, in order to update their personal details. Of course, the website link presented to the user points to the attacker's own website, designed to look like the bank is it masquerading as. On it a form is presented requesting the user's login details. This is then used by the thieves to log on to the real user account and transfer money or make purchases.

Thus, a name, or identity, can be stolen. If a good reputation is attached to the name then the owner

of the stolen name can use it to obtain services destined for that name. The process of authorising those services usually involves some form of authentication of the name to ensure that the agent making the request is the rightful owner of that name. However, the example above illustrates that this system can be subverted when there is asymmetry in the authentication process. While a bank will require its online customers to authenticate themselves (with passwords and PINs), the average user is unaware of the need to strongly authenticate the entity presenting itself as the bank, relying instead on weak forms of authentication based primarily on visual signals (name and logo for example). This system is thus easily exploited.

In the context of this work, a name is simply an object, such as a public key, with an attached reputation, and about which an opinion of trust can be made. Names can thus be transferred between real world entities, effectively transferring reputation. This mirrors brand names in commerce where the reputation of a brand can be purchased by and thus transferred to a different owner [Tad99]. Furthermore, although authentication is part of the recommendation protocol discussed in Chapter 9, the method of authenticating an identity is beyond the scope of our model.

3.9 Chapter Summary

Reputation supports trust by giving the truster information that can be used to assess the prospective trustee's trustworthiness. This information is based on past experiences with the prospect transmitted via word-of-mouth. It is a peer-based social control mechanism, giving its participants a platform upon which to build a cooperative environment.

The significance of reputation has been investigated in various disciplines, such as economics, sociology and computer science. In economics, reputation has even been likened to currency or assets that are potentially tradable.

Stereotype is a form of generalised reputation about a group of agents, rather than towards one agent. By grouping agents together according to their shared attributes, one is able to manage the uncertainty of new encounters by mapping similar attributes to those agents in past experiences and applying the generalised past behaviour into the evaluation of the new prospect's trustworthiness.

In this work, the opinion of a third party about a prospect is communicated in a recommendation. The aggregation of the recommendations from various third parties about the prospect is the prospect's reputation.

Collaborative filtering is one example of how opinions can be shared between a community of agents. The difference between collaborative filtering and reputation systems is that in the former, the emphasis is on similarity of opinions rather than trustworthiness in the source of opinions. The use of reputational information, however, depends on how much we trust the source of opinions that contributed to that reputation.

The identification of sources of opinion, and more generally of agents in a reputation system, requires a naming system to be in place. This is essential as without the ability to refer to an agent unambiguously, reputation information becomes meaningless. Several naming systems are already in place, some relying on centralised name spaces, such as IP addresses, while others remove this requirement and adopt a fully distributed approach, such as SDSI.

In our model, we do not specify how names should be structured as it is beyond the scope of this work. We simply assume that a name is an object for which reputational information can be attached to, and that names can be transferred between owners.

Chapter 4

Trust Models – the Current Landscape

> *We might like to believe that we can trust computers more than we can trust people. However, the bottom line is that, because the computer systems are themselves products of human effort, we cannot expect to trust them either – even if we have gone to enormous efforts to increase trustworthiness.*
>
> *Peter Neumann [Neu92]*

Trust models exist in various areas of computer science, most of them as components in wider areas of application, and a few existing as pure general trust models in their own right. The development of these trust models have been motivated by 1) concerns over security in large scale distributed systems where it is difficult to verify the identity of users before interacting with them and 2) the need to manage vast quantities of complex data and to filter signal from noise.

A number of major areas of research have emerged where trust, and reputation, its corollary, is modeled. The most prevalent is in *computer network security*, going back to when authentication and authorisation protocols were engineered. Concerns about unauthorised access to network services and whether someone on the network is who she says she is have spurred work in creating formal proof tools to verify that trust is granted only to those to whom it is supposed to be granted. More publicly, the wide adoption of *electronic commerce* raised concerns about whether an 'e-commerce enabled' website has the necessary measures in place to ensure that the privacy of their customers is not compromised. This involved trust in the supplier with respect to privacy measures, and also trust in third parties in issuing valid certificates for public key encryption. At around the same time, agent marketplaces, a branch of *Multi Agent Systems* aimed at allowing commercial transactions to be automated, also started examining the issue of trust, in the form of reputation systems. These models attempt to weed out bad agents from good within a community of transacting agents. A related area that emerged is *collaborative filtering* or recommender systems where agents' preferences are matched against each other in order to provide new recommendations. Another area where trust models have been adopted early is *online auctions*. Here, auctions sites such as eBay allow buyers and sellers to rate each other so that good and bad users can be identified publicly. This model has also been successfully implemented in *community portals* like Slashdot for example. Mobile systems research has also considered trust models. This is particularly relevant because of the nature of *mobile systems*, that is users and processes are able to roam across domains governed by different policies and different organisations. The most recent area where trust models have been applied

and discussed, and perhaps the area where trust issues have gained more focus than ever before, is *peer-to-peer* systems. Peer-to-peer technology strives to achieve a truly decentralised network where each node is completely autonomous and transactions do not rely on any central authority. The potential for rampant rogue nodes in this network have prompted system designers to employ at least a minimal trust, or reputation, system to ensure some amount of robustness against malicious nodes.

We will now review the state of the art in how trust is modeled and handled in current research or applied systems and protocols within those areas above. The goal of this chapter is to find out where trust have been explicitly dealt with and whether they match the social properties of trust discussed in Chapter 2. To be exact, we would like to understand how trust handling in these reviewed models can be described under the following headings:

1. **Definition of trust:** which and how the reviewed systems define 'trust'.
2. **Types of trust:** whether the types of trust identified in the social sciences are handled by these systems.
3. **Properties of a trust 'relationship':** what constitutes a trust 'relationship'.
4. **Handling of distrust, mistrust and ignorance:** whether these three concepts are explicitly represented in the system.
5. **Representing trust:** details of concrete representations of trust.
6. **Trust dynamics:** the whats and hows of factors affecting trust.

Table 4.1 shows the computer trust models reviewed. Each item represents a proposed model developed either as a self-standing research project or as a module in a larger application. The column titled 'Key' contains the keys that we will use as reference to the models in this dissertation and the 'Description' describes the works briefly. A taxonomy of the reviewed work is shown in Figure 4.1.

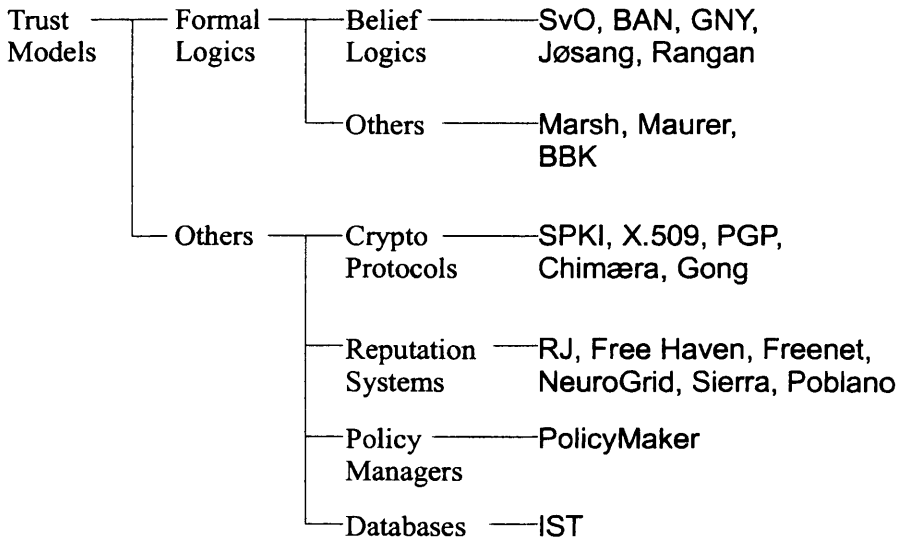


Figure 4.1: Taxonomy of trust models.

Key	Description
BAN	Formal logic for reasoning about the correctness of cryptographic protocols [BAN90].
BBK	Model for finding trustworthy certification chains. [YKB93, BBK94].
Chimæra	A proposed extension to an early X.509 work for adding 'weights' to certification paths [TH92].
Free Haven	A peer-to-peer data storage and distribution system that emphasises anonymity and data publisher control. [Din00, Sni00, frea].
Freenet	Similar to Free Haven, a distributed data storage system, but with emphasis on availability, privacy and security [CMH+02, freb].
GNY	Formal logic for reasoning about the correctness of cryptographic protocols, as in BAN, but includes more extensive reasoning about messages and their semantics [GNY90].
Gong	A shared control scheme for distributing trust in key generation to a set of authentication servers [Gon93].
IST	A relational database extension to keep track of sources of tuples in tables in order to asses their credibility as information sources [SJ93, JS93, JS94].
Jøsang	A subjective logic for belief that can be used to model and rationalise beliefs relating to levels of trust in a system [Jøs97a, Jøs98b, Jøs96, Jøs97c, Jøs97b, Jøs98a, JK98, Jøs99].
Marsh	A formal model of trust for artificial agents [Mar94a, Mar94b].
Maurer	Deterministic and probabilistic models for certification [Mau96]. If the need arises to discriminate Maurer's deterministic and probabilistic models, we will use Maurer-D to refer to the former Maurer-P for the latter.
NeuroGrid	A peer-to-peer keyword based search environment that claims to mimic 'word of mouth' in real societies. [Jos02, web].
PGP	An email and file encryption software [Zim94].
Poblano	Proposed trust model for the JXTA peer-to-peer platform. [CY01, Mic04, jxt].
PolicyMaker	A 'trust management' tool for implementing authorisation policies based on public keys and an interpreted implementation language [BFL96].
RJ	A reputation-based platform for artificial buyer and seller agents [Ras96, RJ96].
Rangan	A 'possible worlds' formal model for trust [Ran88].
Reagle	An analysis of various trust mechanisms [Rea96].
Sierra	Reputation management framework used to support OpenPrivacy, a user profile management project. [Ope].
SPKI	Acronym for 'Simple Public Key Infrastructure', which is an authorisation protocol for services in a distributed system [E+99].
SvO	A logic for reasoning about cryptographic protocols, based on the combination of previously defined logics of Abadi and Tuttle (AT) [AT91], BAN and GNY [SvO94].
X.509	An ITU-T standard for a directory authentication framework. The version reviewed here is X.509 version 3, which is the current version at the time of writing [ITU97, Bra97].

Table 4.1: List of reviewed computer science work on trust.

4.1 Definition of Trust

As we have seen in Chapter 2, the task of defining trust is a real challenge even to social scientists themselves. Those that attempted it have found that their definitions often do not completely agree with one another. It is no wonder that concrete definitions of trust are found wanting in areas of computer science where the concept of trust is concretely represented. Nevertheless, the elusive nature of the concept of trust requires that any application that uses, or more importantly, relies on trust, define it explicitly. Failure to do so will result in poorly understood and ambiguous application or data semantics that may lead to vulnerability in the system. Thus, it is disappointing to discover that not all authors of the reviewed models attempted to answer the question "what is trust?", as applied in their work. It can be said, however, that in general the notion of trust in systems such as cryptographic security protocols involves an expectation that a trustee will behave in a certain way.

We summarise the definitions in Table 4.2 below, mimicking McKnight and Chervany's approach [MC96] by comparing the different properties inherent in the definitions. Explanations for the tables columns then follow. Models that are not listed in the table did not provide a definition of trust in any of their documentation.

Key	Term Used			Perception		Action-Oriented	Scope of Trust
	"Trust"	"Trusted"	"Trustworthy"	Belief	Property		
BBK	•			•		•	Authentication
Free Haven	•			•		•	Protocol execution
IST	•				•		General
Jøsang	•			•		•	General
Marsh	•			•			General
Poblano	•			•			Source of relevant data
PGP		•		•		•	Key signing
Reagle	•			•			General
X.509	•			•		•	Certificate Issuance

Table 4.2: Trust definitions for computer systems.

Notes for Table 4.2:

1. Reagle regards trust as some 'degree of validity of a statement'. This is similar to Jøsang's representation of trust as a general belief about something.
2. Jøsang separates definitions for passionate (agents who can choose not to be malicious) and rational (agents without the notion of malice) agents.
3. Only PGP's 'trust in introducer', and not its trust in key validity, is action-oriented.
4. BBK gave different definitions for 'Direct' and 'Recommender' trusts.

4.1.1 Terms Used

In the Computer Security Reference Book [JH94], the meanings of 'trust', 'trustworthiness' and 'trusted' were defined. This is important because some users may find these terms carry different semantics, as discussed in §2.5. For example, Alice may find that Bob is 'trustworthy' (with respect to something), but may not actually 'trust' him under certain unforeseen conditions, like when the

perceived risk in trusting Bob becomes too high. Although some of the models surveyed above used more than one of these three terms, their definitions were not explicitly put forth.

4.1.2 Subjective Belief or Objective Property

All of the models in the table defined trust subjectively, or as some form of belief, except IST. Representing trust as a subjective belief matches closely the definitions of trust found in the social sciences. IST, however, defined trust as the "reliability" of an agent, based on past experiences with the agent's statements that may be true or false (details will be discussed in subsequent sections). This can also be seen as a probability value based on samples of the agent's past behaviour.

4.1.3 Action-Orientation

This property shows us whether trust was defined as something related to expectations about the trustee's future actions. Of the above, only the definitions of X.509, Free Haven, PGP, Jøsang and BBK had this property. Marsh applies trust to the general notion of an 'outcome', and Reagle considers assertions that are "valid in the real world".

4.1.4 Scope of Trust

This is to discover whether the definitions include any specific scope for trust, or whether trust was defined as a general notion. X.509 and PGP defined trust within the scope of signing public-key certificates and BBK considers all actions involved in an authentication process as its scope. In Free Haven the scope of trust is limited to adherence of proper network protocol. It also has a more generally defined 'metatrust' which is about trusting the recommendations of a node. The rest defined trust generally.

4.2 Types of Trust

Three types of trust were identified by social scientists (see §2.2), namely Interpersonal, System Trust and Dispositional Trust. In this section, we will attempt to find these three types in our surveyed models. We will also find out if they include the situational or cross-situational dependencies of the three types of trust mentioned. Our findings can be summarised in Table 4.3 below.

The table shows that most models only define Personal/Interpersonal trust, which may not be surprising as it deals with direct trust in another agent or entity.

Some models define specific situations for their trust, e.g. X.509 specifies trust only in context of creating reliable certificates. Others are Gong (authentication), PGP (key introduction), RJ (buying something from a seller agent), NeuroGrid (recommending relevant documents for a given keyword) and Maurer (key and transitive introduction). Gong, in addition to the specific situation of authentication, also allows trust parameters to be tweaked for each authentication process, allowing further refinement of situations, based intuitively on a user's judgement. Freenet, Free Haven and Poblano defines trust within the situational context of making data available when required and following proper network protocols.

Key	System (S)	Dispositional (XS)	Interpersonal (S)
Jøsang	○	○	○
Gong	•		•
Sierra	•		•
Rangan	○		•
PGP		•	•
Marsh		•	•
Reagle		•	•
X.509			•
Chimæra			•
SPKI			•
RJ			•
IST			•
Maurer			•
BAN			•
GNY			•
SvO			•
BBK			•
PolicyMaker			•
Freenet			•
Free Haven			•
NeuroGrid			•
Poblano			•

Table 4.3: Trust typology in current trust models. The correct type of situational dependency for the particular trust type is shown in brackets in the table heading, with (S) for Situational and (XS) for cross-situational dependencies respectively. • indicates support for that trust type. ○ indicates the potential to handle that trust type, although not explicitly defined in the model as such.

The rest allow arbitrary situational contexts to be defined. SPKI allows arbitrary authorisation classes to be delegated using its authorisation field, or ⟨tag⟩. In IST, the situational dependency is also called the agent's 'expertise', which depends on the fields in the relational database that the agent contributed information to. These database fields can be arbitrarily defined. Marsh provides variables to reference specific situations in his formalism, thus allowing arbitrary situations to be considered. BAN, GNY and SvO provide logics for reasoning about belief in an agent's 'jurisdiction'. An agent may be believed to have jurisdiction over (or have authority on) the content of a message. Therefore, the content of the message itself determines the situation which is relevant to the perceived jurisdiction that the agent presides over. BBK allows arbitrary situations, or 'trust classes', to be defined for each trust relationship. In PolicyMaker, trust in keys is defined according to constraints in the user's policy, written in the PolicyMaker language, which allows arbitrary situations to be defined. Reagle merely mentions that experts in the area of X are more believable with respect to X. Sierra merely defines trust with respect to whether a peer is reliable or not. Exactly what it is reliable for is left open to implementation.

Jøsang's subjective logic of belief is general enough such that different trust types and situational dependency can easily be accommodated. This is due to the generality of the concept of 'belief', which Jøsang's logic was designed to handle. System trust, for example, can be modeled by "belief in the trustworthiness of the system" while Dispositional trust can be modeled by, for example, "belief that agents in situation X are generally untrustworthy".

Rangan's belief logic, with a possible-worlds semantics, also allows one further trust type to be derived, although not explicitly defined, i.e. System trust. For example, the belief that an entity "j will be punished if he lied" (a trust in the judicial system) can be represented by the belief statement B_jQ in Rangan's logic and assigning Q = "j will be punished if he lied" (see [Ran88] for details). However, Dispositional trust cannot be represented by Rangan's logic because Dispositional trust reflects "degrees of willingness", and, since there is no notion of 'levels of trust' in Rangan, it cannot be represented. Dispositional trust represented in Rangan's logic will carry either disposition of 'everybody is trustworthy' or 'nobody is trustworthy', which is very limiting.

4.3 Properties of Trust

Here, we are interested in the properties of trust relationships, or trust beliefs of agents, as applied in the areas surveyed. Some overlap with our survey of trust definitions above will be encountered, especially with respect to whether trust is represented as a subjective belief or an objective property of an agent. However, we are more concerned with the details of these representations here, rather than just whether they were defined as having either of those representations. Furthermore, we will include here those models that did not produce a concrete definition of trust, but have implicitly used these representations in their work. Similarly, situational dependency will be looked at in more detail here, identifying how situational dependency is applied.

In addition to the above, we are also concerned their ability to handle uncertainty, manifested as their ability to represent *levels* of trust . This property will be looked at in more detail in §4.5 later in this chapter. Lastly, we will see whether each surveyed trust model allows transitivity. As before, we first present a summary of our findings, shown here in Table 4.4.

4.3.1 Objective Representations

Of the surveyed trust models, three represented trust objectively. Chimæra's trust model is based on a directed graph of certification authorities (CAs) and weighted certification paths. The 'trust degrees' are propagated via neighbouring CAs and used at 'face value' without any translation. This arbitrary weighting scheme actually increases uncertainty in the calculation as the semantics of the values used is unclear. Furthermore, the absence of the source that defined those values makes it difficult to ascertain whether 1) the value is meaningful and 2) whether it can be relied upon. In fairness, the authors did voice this concern in their paper and envisaged that future modifications to Chimæra will include propagated *characteristics* of CAs, e.g. the trust they attach to their neighbours, which could be used by the local policy. In effect, Chimæra proposes to include a description of its local used metric besides the trust value. This will allow a remote CA to convert this trust metric in accordance with both the local and the remote one" [TH92]. Unfortunately, no further details followed this interesting proposal.

IST's objective model is stronger as it *can be* based on observable facts. The value of an agent's trustworthiness, or reliability, is based on what the agent contributes to the database as 'fact', verified by observations of whether that 'fact' is true or false. By keeping a database on an agent's past contributions and whether each contribution is true or false, as observed by the database, the trust value can be fairly consistent from one observer to another. This is subject to how the observed fields

Key	Subjective	Objective	Binary	Degrees	Transitivity	Constraints
BAN	•		•		•	•
BBK	•			•	•	•
Chimæra		•		•	•	
Freenet	•		•		•	
Free Haven	•			•		
GNY	•		•		•	•
Gong	•			•		
IST		•		•	N/A	•
Jøsang	•			•	•	•
Marsh	•			•		•
Maurer-D	•		•		•	•
Maurer-P	•			•	•	•
NeuroGrid	•			•		•
PGP	•			•	○	•
Poblano		•		•	•	○
PolicyMaker	•			○	•	•
Rangan	•		•		•	•
RJ	•		•			
Reagle	•			•	•	•
Sierra	•			•	•	
SPKI	•		•		•	•
SvO	•		•		•	•
X.509	○	○	•		•	•

Table 4.4: Properties of existing trust models. A hollow circle (○) indicates properties that require further comments, which will be elaborated in the subsections that follow.

are defined, i.e. the more concrete the observed facts are (e.g. "there is a coin under the rock") then the more reliable the trust values.

Poblano's model is similar to Chimæra's in that trust values are taken at face value. The difference is that the source of trust values in Poblano is known. In a trust propagation chain the trust value of a node is the opinion of the node prior to it. However whether the source of the trust value is known or not has little impact on the trust model because of the fact that the values are taken at face value.

4.3.2 Subjective Representations

The models that consider trust as a subjective construct can be broken down into two groups. The first group consists of those models that allow an agent to make subjective trust statements (e.g. BBK, Maurer, Sierra, PolicyMaker and X.509) or assign subjective trust 'values' (e.g. BBK, Freenet, Free Haven, Poblano, NeuroGrid, Gong, Marsh, PGP, RJ, Sierra and SPKI) *based on* the agent's beliefs. An interesting, albeit confusing, feature of Poblano is that although trust values are subjective 'opinions' of individual agents, the values become objective 'properties' of trustees as soon as they are conveyed to other agents. When Alice tells Bob that Cathy is untrustworthy, Bob simply takes Alice's word for it, without trying to determine if Alice herself can be trusted about that statement.

The second group (BAN, GNY, Jøsang, Rangan and SvO) consists of those models that explicitly manipulate 'belief' constructs. These models, all of them formal logical frameworks, allow the reasoning about belief itself. Of the models mentioned in this group, only Jøsang represents belief

as a probability estimate while the rest represent belief with statements of the form '*A believes x*', which is usually used in conjunction with inference rules, as in one of SvO's 'believing' axioms, which says that "a principal believes all that logically follows from his beliefs" [SvO94, modified notation]:

$$A \text{ believes } x \wedge (x \Rightarrow y) \Rightarrow A \text{ believes } y$$

In addition, the logics of BAN, GNY, SvO and Jøsang provide for the reasoning of n-level beliefs, or beliefs about beliefs (about beliefs, ad infinitum). For example, in addition to forming beliefs about the trustee's trustworthiness, it may sometimes be necessary to ascertain if the trustee himself believes that we are trustworthy - a classical Prisoner's Dilemma problem, as discussed in §2.8.2. Thus we are confronted with the problem of processing second order beliefs: beliefs about others' beliefs about us. One can imagine this problem expanding to involve arbitrary levels of belief when we incorporate more players into the game. GNY argues why reasoning about more than one level of belief can be important in reasoning about cryptographic protocols specifically which can be generalised to the meaning of protocol messages in general [GNY90]:

> *... since each principal expresses his beliefs by sending messages, we choose to interpret the beliefs on the sender's part as the preconditions for a message to be sent at all. ... A message of the same form may carry different, context dependent, meanings in different protocols. Having believed a message to be genuine, a recipient can choose to believe the sender's beliefs, if he trusts the sender's honesty and competence in following the protocol specification. In other words, since we do not require the universal assumption that all principals are honest and competent, we should reason about beliefs held by others based on trust on different levels.*

4.3.3 Binary Trust or Degrees of Trust

Social trust, as we know at this point, is a certain degree of belief. What we are concerned with here is whether this property of trust is reflected in the surveyed models.

BAN, Freenet, GNY, Rangan, RJ, SPKI, SvO and X.509 considers trust to have an 'all-or-nothing' property, i.e. there is either complete trust, or no trust at all. Of those that modeled degrees of trust, Chimæra, Free Haven, Gong, Maurer-D, PGP and Poblano represented them as discrete levels while BBK, Jøsang, IST, Marsh, Maurer-P, NeuroGrid, and Sierra used real numbers between 0 and 1 (see Table 4.7). NeuroGrid uses percentages as the trust value, although this can be alternatively represented as real numbers.

In PolicyMaker, trust decisions are really made by the programs to which the user policy delegates to. All PolicyMaker does is to receive queries in the form of "*key(s)* REQUESTS *ActionString*" and correctly pass on the key(s) and *ActionString*s to the applications that handle them. The output of PolicyMaker is a decision of whether the key(s) can or cannot carry out the actions requested in the *ActionString*. Therefore, the graininess of trust really depends on the application that verifies the *ActionString*, and not PolicyMaker.

Further details of how the surveyed models represent trust levels will make this subsection overly

long, and therefore we continue our discussion of trust representation in §4.5 below.

4.3.4 Transitivity

A common misconception is that trust is transitive. This is not necessarily so, as we have seen in §2.1 and also shown formerly by Christianson and Harbison [CH96]. Nevertheless, the prevailing assumption in the models surveyed is that trust *is* transitive, although the authors themselves may admit to trust's non-transitivity nature, e.g. as Jøsang states in [Jøs99], in relation for supporting recommendations in his model:

> *The recommendation operator can only be justified when it can be assumed that recommendation is transitive, or more precisely that the agents in a recommendation chain do not change their behaviour (i.e. what they recommend) as a function of which entities they interact with. However, ... this can not always be assumed, because defection can be motivated for example by antagonism between certain agents.*

As hinted in Jøsang's quote above, the inclination to model trust as having a transitive property stems from the need to support the propagation of beliefs about trustworthiness from one agent to another. For example, if Alice trusts Bob and Bob trusts Cathy (with respect to something), it may seem intuitive that Alice may be likely to trust Cathy also upon Bob's recommendation, or upon Alice's observation of Bob's trusting actions towards Cathy. A model that has this behaviour is much more useful than one where beliefs about another's trustworthiness cannot be gained from a third party.

However, a common error is to assume that transitivity is based solely on the trust relationships per se. In social relationships, this apparent transitivity is derived from a much larger knowledge base, which includes familiarity with the third party, experiences of the truster, his disposition and many other factors that affect trust, as discussed in Chapter 2. There is also an added danger in models that view trust as binary (complete trust or no trust at all) and transitive at the same time (as in BAN, GNY, Rangan, SPKI, SvO and X.509) – a malicious agent may take advantage of this and manipulate a weak agent to gain complete trust in another agent who trusts the weak agent completely. Trust relationships in the UNIX system are one such example [GS96].

There is a user parameter called CERT_DEPTH in PGP that defines the maximum allowable length of certification chains, suggesting an assumption of transitivity. However, PGP's algorithm for evaluating key validity only takes into consideration signatures from certificates directly trusted (completely or marginally) by the user. This means that all certificates in the chain must be directly trusted by the user too. In effect, there can ever only be acceptable certificates that are one level below the user, i.e. directly trusted by the user. Therefore, this makes the CERT_DEPTH parameter in PGP redundant, and all trust relationships in PGP non-transitive.

IST does not model transitivity because its domain (relational databases) excludes any consideration for belief propagation or any kind of information exchange between agents.

Transitivity is a very powerful yet fragile concept for trust. It bridges the gap between known and unknown agents and allows complex relationships to emerge. However, a chain is only as strong as its weakest link and if one is to allow transitivity in the trust model, its implications must be fully

investigated. To illustrate its fragility, we examine how Poblano manages transitivity.

In Poblano, a keyword search is forwarded down a chain of agents until it reaches an agent that has access to the file. Each agent in the chain, apart from the requesting agent, is attached a trust value and each trust value is assigned to its corresponding agent by the agent prior to it (called the recommending agent). The trust value represents how much the agent is trusted by its recommender to recommend another agent to grow the chain. This is true except for the trust value of the last node which represents how much the recommending agent trusts it with respect to returning a file that will match the search keyword(s). So in the example chain in Figure 4.2 Alice needs to do a search on keyword k. She sends this request to R_1 because she trusts R_1 with trust value $t(R_1)$ to find or recommend other agents with respect to returning something that will fulfill her search request. Similarly R_1 sends this down the chain to R_2 until the request reaches Bob who has the matching file. This file is then returned directly to Alice by Bob along with the propagation path information.

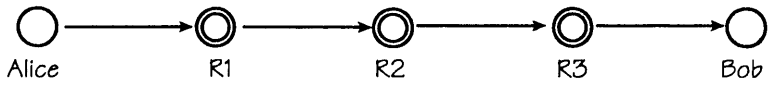


Figure 4.2: Example simple P2P search path.

at first glance, this chain may appear valid, with intermediate agents and endpoint agents trusted with respect to the appropriate trust contexts. The problem arises in the implementation of Poblano specifically when Alice tries to calculate the final trust value on Bob based on the trust values along the chain. In Poblano the values are simply averaged according to Equation (4.1) below, giving Alice's final trust in Bob as:

$$\frac{1}{4n}\left(\sum_{i=1}^{n} t(R_i)\right) \times t(Bob) \tag{4.1}$$

The first statement is divided by $4n$ to normalise the summed recommended values back into the trust value scale of $[-1..4]$. Now if R_2 is a malicious agent the formula above will simply absorb this fact and continue to use the chain when the chain has clearly been compromised. In fact, and worse still, Poblano actually recommends that recommenders in the chain are simply skipped and ignored in the calculation and that the formula should only sum up trust values in the chain that are greater or equal to one!

This inconsistent implementation of a trust model illustrates lack of understanding in how trust functions in society and suggests the need for a better understanding of social trust and careful analysis of implementations of trust models.

Poblano's trust model is based on a very early version of our work, published in a workshop paper [ARH97a]. However, since this initial and in retrospect, somewhat naive attempt to capture the nature of trust, we have made significant improvements.

In Sierra, transitivity is also supported but this time instead of averaging the model takes a risk-averse stance and uses the lowest value in the trust chain. Prior to selection, these values would have been reduced according to their distance from the head of the chain using values is a *Node Distance*

Distrust Table that indicates how much to reduce the trust value by depending on the number of hops it is from the head. However, the values in the *Table* are defined on an ad hoc basis by each individual user to say how much he trusts the n-th degree of a chain. It reflects more on the user's trust in the System (the network) rather than the agents in the node.

4.3.5 Constraints

Constraints on trust relationships represent the conditions when the trust relationship is valid. All the models surveyed, except Chimæra, Gong and RJ provide some way of asserting constraints on trust relationships. Those that allow constraints differ in the range and format of constraints allowed. This is shown in the taxonomy in Figure 4.3.

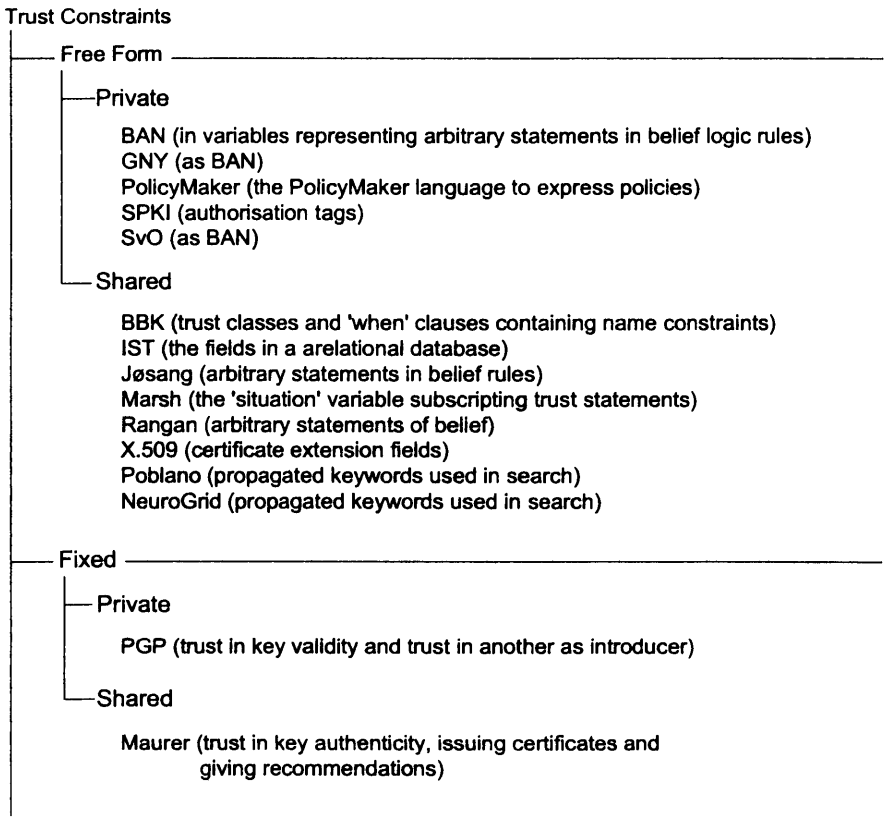


Figure 4.3: Taxonomy of constraint types for trust relationships. Under 'free form', the implementation methods for each model is given in brackets. Under 'fixed', the fixed constraint types are given in brackets. Please refer to their corresponding references as given in Table 4.1 for details.

In the taxonomy in Figure 4.3, constraints are divided into 'free form' and 'fixed' types. Models that support free form constraints allow agents to define their own constraint types freely. No specific constraints are imposed on what constraints can be formed, apart from perhaps the limitations of the language that constraints can be expressed in. For example, in PolicyMaker, constraints can only be expressed in the PolicyMaker language (examples of its vocabulary and grammar are given in [BFL96]. This, for instance, allows sophisticated or even esoteric constraints to be defined, such as "give agent X the combination and key to the safe if and only if X is an immediate family member over the age of 35 without any criminal record and X promises to return the key by midnight on the

day the safe's key is handed to X". Free form constraints make the model more general and flexible. However, the semantics of free form constraints may become ambiguous to agents other than the one who defined the constraint, i.e. when used as shared free form constraints. This problem is compounded by large systems that encompass multiple cultural domains. Therefore, shared free form constraints must be used with caution and only when strict discipline has been observed regarding documentation of their semantics.

X.509 implements shared free form constraints in a *certificate extension* field. An extension can include information on constraints about the validity of the certificate (e.g. the X.509 defined 'Private key usage period' field) or information about trust in the subject of the certificate (e.g. the X.509 defined 'Name certification path constraint' field). Extension fields can be flagged critical or non-critical and X.509 specifies default behaviours with respect to these flags as follows [ITU97]:

> *If an extension is flagged critical and a certificate-using system does not recognize the extension field type or does not implement the semantics of the extension, then that system shall consider the certificate invalid. If an extension is flagged non-critical, a certificate-using system that does not recognize or implement that extension type may process the remainder of the certificate ignoring the extension.*

Reagle suggested that agents who are experts in a particular area should be trusted more than others who are not. Although he is suggesting the use of constraints in different words, we will not include it here because it is too general and he did not specifically indicate how this should be modeled.

A note on 'subtypes' of constraints

Consider the statement "I trust Alice to do Y on condition that Z". It may be intuitive to say that there are two types of constraints present in that statement. The first is Y, i.e. what we are trusting Alice *for*. The second is Z, i.e. under what *conditions* do we trust Alice for Y. Thus, for completeness, it may make sense to model both in a trust model. However, it can also be argued that Y and Z are just constraints expressed at different granularities. Looked at in this way, it is natural to extend the notion into arbitrary levels of granularity. This view makes it difficult to define a bounded number of subtypes – how many subtypes does a trust constraint have? The answer is that it depends on the application domain. In the model presented in this work we model a single 'constraint' type. We believe that this is sufficiently general such that it can be used to support arbitrary levels of granularity in constraints when required, yet simple enough to be used for systems with very coarse granularities of constraint expression.

4.4 Handling Non-Trust: Distrust, Mistrust and Ignorance

In this section, we will discover whether the models explicitly consider *distrust*, or finding another agent untrustworthy, *mistrust*, i.e. when a trusted agent reneges and *ignorance*, i.e. when an agent whose trustworthiness is unknown is encountered. Table 4.5 below summarises our finding in this section.

Note that the handling of mistrust is not applicable in static models, or models that do not have a feedback loop that will allow it to make use of experiential data to make future trusting decisions.

Key	Distrust	Ignorance	Mistrust
BAN	•		N/A
BBK	•		N/A
Chimæra			N/A
Freenet			•
Free Haven			
GNY	•		N/A
Gong			N/A
IST		•	•
Jøsang	•	•	N/A
Marsh	•	•	•
Maurer			N/A
NeuroGrid			
PGP	•	•	N/A
Poblano	•	•	
PolicyMaker			N/A
Rangan	•		N/A
RJ	•		•
Reagle	•		N/A
Sierra	•		
SPKI			N/A
SvO	•		N/A
X.509	•		N/A

Table 4.5: Support for distrust, ignorance and mistrust.

These static models usually delegate such decisions to agents outside the model, e.g. by allowing the user to adjust an introducer's trust degree in PGP. There are eight dynamic models in our survey: Freenet, IST, Marsh, RJ, NeuroGrid, Poblano, Sierra and Free Haven. However, only the first four supports mistrust.

IST, for example, separates 'predicted' and 'observed' tuples in its relational database model. An observation is compared with the prediction, and then the predicted data is moved into either the 'fact' or 'hoax' partition of the observed database. The latter can be considered also as instances of mistrust. Marsh takes a different approach, by providing a trust updating algorithm which reduces a trust level by a certain 'level of modification' for each bad experience. In RJ, a mistrust is followed by future non-interaction with the mistrusted agent as well as future non-recommendation. This method is also used in Freenet, where mistrusted nodes in the network are simply removed from the user's search routing table. There is no explicit support for negative recommendation in either Freenet or RJ.

Ignorance[1], or the absence of opinion about another agent's trustworthiness, is included in five of the surveyed models. IST, Marsh, Poblano and PGP uses a single value to represent ignorance about an agent's trustworthiness. Jøsang, however, represents *degrees* of ignorance[2]. This may seem counter-intuitive as ignorance is merely the absence of an opinion, and representing 'degrees of absence' may not seem sensible. However, he justifies this [Jøs97b] by positing that:

[1] Defined on page 49

[2] Jøsang changed the use of the term *ignorance* in his earlier work [Jøs97b, Jøs97a] later into *uncertainty* [Jøs98b, Jøs98a, JK98, Jøs99].

> ... *the human mind maintains belief and disbelief simultaneously as quasi-complimentary, and that when both beliefs become weaker, due to insufficient evidence, ignorance increases.*

Thus, in Jøsang, an opinion is a triplet containing components of belief, disbelief and ignorance, all of which may contain non-zero values at the same time. The sum of the three components always equal to one. Further details of this representation will be given in §4.5 below.

Of the remaining models that do not handle ignorance explicitly, ignorance is sometimes defaulted to distrust. BBK, for example, says that "a principal is assumed not to trust another unless there is an explicit expression specifying it does" [BBK94]. This is also true of all the remaining models as there is usually an algorithm or logic rule that requires a positive trust expression to be satisfied before an agent can be considered to be trustworthy enough for interaction. Therefore, although they do not explicitly state default assumptions as BBK did, we will assume that BBK's assumption also holds implicitly in the other models that do not handle ignorance explicitly.

Reagle did not include ignorance in his discussions about a general model of trust, therefore we did not include it in the table.

Upon an encounter with a distrusted agent, the agent, or its messages, is either completely ignored, or presented to the user with an indication of its untrustworthiness, leaving it up to the user to decide further action. The first approach is taken by the majority of the surveyed models which handles distrust explicitly, i.e. BAN, BBK, GNY, PGP, Rangan, RJ, SvO, Poblano, Sierra and X.509, while Jøsang and Marsh adopts the latter. Thus, it can be said that the Jøsang and Marsh models are more sympathetic towards agents that are considered untrustworthy, compared to the other eight models that chose to ignore them. However, all the models that support distrust are still considered pessimistic as none of them specifically give untrustworthy agents the chance to prove them wrong through additional experience. This may prevent users of these models from uncovering fruitful avenues of interaction with potentially valuable agents that are considered untrustworthy, due perhaps to early one-off experiences with them.

4.5 Trust Representations and Their Semantics

Here we will look in detail at how trust is represented as a concrete data structure in the models surveyed. We first look at models that represent trust as binary values and them look at those that incorporate varying trust degrees. The reader can refer to Table 4.4 for models in either of these categories of representation. Additionally, we will also look at the semantics of their representation. An interesting point to note in this section is the semantics of recommendation trust in other agents, i.e. the trust in another agent to recommend other agents within a certain context. For this type of trust we try to discern whether the value represents a 'context similarity measure', or an actual trust degree assigned to that agent. This impacts assumptions on the transitivity of trust; that is if the value represents a context similarity measure, then trust need not be assumed to be transitive, unlike the requirement for defining the values as a measurement of recommendation trust.

4.5.1 Binary Representations

A binary trust representation allows complete trust in another agent or no trust at all. This is modeled by BAN, GNY, SvO, Rangan, RJ, SPKI, Freenet and X.509. Trust in another agent is expressed by making specific trust assertions. These assertions vary in form. One way is to make assertions, of the form "I trust Bob with respect to X" or "Alice trusts Bob with respect to X". Assertions are made in this fashion in BAN, GNY, SvO, Rangan, RJ and X.509, although the syntax may differ. A trust relationship can also be represented implicitly, as in SPKI and Freenet. SPKI does not support specific trust constructs, but allows an agent to delegate authorization to another agent, implicitly forming a trust relationship. A summary of how binary trust is represented, and their semantics, is given in Table 4.6 below. Similarly in Freenet, the presence of an agent in its routing table indicates trust in that agent.

Binary trust representations are simple constructs and allow unambiguous implementations. The concept is, nevertheless, rather restrictive because users are forced to choose between trusting another agent completely or not at all. The ability to handle degrees of trust will allow users to proceed in situations where the amount of trust in another agent is not complete, but sufficient for the situation concerned.

Key	Syntax	Semantics
BAN	*P believes Q controls X*	Agent P believes that agent Q is an authority on matters concerning X, and therefore should be trusted on matters of X.
Freenet	Presence of a forwarding node (trustee) in the routing table	The owner of the routing table believes that the trustee can be relied upon to find the required file.
GNY	$P \models Q \models\!\!\Rightarrow X$	As BAN, i.e. P believes Q has jurisdiction over X.
RJ	Agent simply utters a statement in the logical form of "I trust P".	The agent trusts agent P.
SPKI	By issuing a SPKI certificate containing (Issuer, Subject, Delegation, Authorisation, Validity)	The Issuer asserts that Subject is granted Authorisation, valid until the time given in Validity.
SvO	*P believes Q controls X*	As BAN and GNY above.
X.509	Signing, or 'issuing', an X.509 certificate.	The signer asserts that the subject's credentials are correct as given in the certificate, and that the key should only be trusted according to any trust-related information in the certificate, as provided in any certificate extension.

Table 4.6: Binary trust representations.

4.5.2 Representing Degrees of Trust

Degrees of trust are represented as either discrete levels or a variable taking on real numbers, or continuous levels of trust. Table 4.7 summarises how they are represented in our surveyed models.

Key	Nature of Trust	Disc/Cont	Value Range	Semantics of values
BBK	Direct	C	0.0..1.0 exc. 1	Objective probability.
	Recommendation	C	0.0..1.0 inc.	Degree of similarity.
Chimæra	Direct	D	0..∞	Undefined (intuitive).
Free Haven	Direct	D	0..∞	Undefined.
	Meta Trust	D	0..∞	Undefined.
Gong	System	D	0..∞	Threshold value.
	Direct	D	0..∞	Relative weight.
Jøsang	Opinion=$\{b, d, i\}$	C	0.0..1.0 inc.	Intuitive degree of belief, disbelief and ignorance of opinion.
IST	Direct	C	0.0..1.0 exc.	Objective probability.
Marsh	Basic	C	-1.0..1.0 inc.	Intuitive.
	General	C	-1.0..1.0 inc.	Intuitive.
	Situational	C	-1.0..1.0 inc.	Intuitive.
Maurer-D	Recommendation	D	1..∞	Meaningful values.
Maurer-P	Recommendation	C	0.0..1.0 inc.	Subjective probability that statement is true. Intuitive.
NeuroGrid	Recommendation	C	0.0..1.0 inc.	Objective probability that search result is relevant.
PGP	Dispositional	D	0..∞	Threshold value.
	Direct	D	(see Table 4.8)	Meaningful values.
Poblano	Recommendation	D	-1..4	Meaningful values (see Table 4.9).
Sierra	Interpersonal	C	-1.0..1.0 inc.	'Poor Quality' to 'High Quality'.
	Self Trust	C	-1.0..1.0 inc.	Confidence in own rating ability.
	System	C	0.0..1.0 inc.	'Weight' given to an edge in trust chains.

Table 4.7: Representations of Degrees of Trust. Disc/Cont indicates whether the values are discrete (D) or continuous (C). The value range is shown as inclusive (inc.) or exclusive (exc.).

Discrete trust levels

Both Gong and PGP uses a 'threshold' semantics for some of their trust value variables. The notion of threshold has been suggested also by social scientists with respect to social trust (see §2.4.3). In Gong, the user specifies the threshold for the number of compromised authentication servers, in order for the protocol to succeed. This is achieved by a threshold function $ft_n()$ which produces n shadows of x such that x can only be reproduced from at least t number of shadows and no less. This, although not explicitly modeled so, is Gong's representation of System Trust, i.e. the user's trust in the system comprising n number of servers, expressed as a threshold number. In PGP, the trust in key validity[3] depends on two parameters; the number of signatures from completely trusted introducers, and the number of signatures from marginally trusted introducers, both represented in PGP's `COMPLETES_NEEDED` and `MARGINALS_NEEDED` global parameters respectively. For example, if the users set `COMPLETES_NEEDED` to three, then at least three completely trusted signatures are required on the certificate before the public-key in the certificate is deemed to be *completely* valid. If neither parameter is satisfied, but at least one signature of either level is present, then the key is considered marginally valid (or marginally trusted). These parameters allow the user to change the user's general level of confidence in all introducers in general. Thus, this can be seen as PGP's representation of Dispositional Trust.

[3] A *valid* public key is a key that is believed to be strongly bound to its ID.

An alternative way of representing discrete trust values is to have each value carry its own semantics. PGP supports this for trust in both key validity and trust in introducers and Poblano uses a similar range for all its trust relationships. The different levels of trust are given in Table 4.8 and Table 4.9 for PGP and Poblano respectively.

Trust Context	Trust Value	Semantics
Key Validity	*undefined*	We are unable to say whether this public key is valid or not.
	marginal	This public key *may* be valid.
	complete	It is believed with confidence that this public key is valid.
Introducer Trust	*don't know*	The user is ignorant of this public key's trustworthiness.
	untrustworthy	This public key should not be trusted to introduce another, and therefore all signatures associated with this key should be ignored.
	marginal	This public key can be trusted to introduce another key, but it is uncertain whether it is fully competent to do that.
	full	This public key is always trusted to introduce another public key.

Table 4.8: PGP's trust levels.

Level	Semantics
-1	Distrust
0	Ignore
1	Minimal trust
2	Average trust
3	Good trust
4	Complete trust

Table 4.9: Poblano's trust levels.

Individual semantics for each trust level is also adopted by Maurer-D. However, the difference is that instead of providing strata as does PGP, Maurer defines the trust levels recursively. In Maurer-D, A's trust of 'level i' in another agent X, expressed as $Trust_{A,X,i}$, means that A trusts X with respect to recommending trust chains up to level i-1.

Although it is useful to allow an agent to be trusted for recommending chains of arbitrary length as Maurer-D does, it illustrates a lack of understanding about the semantics of recommendation trust. To show why this is so, consider the two statements of trust below:

"I trust my G.P. to recommend me *a good heart surgeon.*"

and

"I trust my G.P. to recommend me *someone from Guys Hospital*
who can recommend me a good heart surgeon."

According to Maurer-D, the two trust statements above can be captured by the general trust assertion

$$Trust_{I,myG.P.,i}$$

Although it is clear that we are trusting the G.P. with different things in each sentence above, this is not reflected in Maurer-D's general trust assertion. Therefore, it is misleading to allow such general trust assertions to mean that a trust of level i indicates that the subject can be trusted for *all* recommendation levels up to level i, because there is a clear distinction of semantics for each level of recommendation, as illustrated by the two sentences above. In other words, different values of i in the general trust assertion gives the assertion different meanings. This is not reflected in Maurer-D's model.

Therefore, when an agent is "trusted to make recommendations of level i", the correct assumption should be that the agent is trusted to make recommendations of level i, *and only* i, no more and, equally importantly, no less.

Another discrete level semantics is supported by Gong, i.e. that of relative weighting. In Gong, servers that are deemed more trustworthy are given more 'votes'. Each shadow is considered one vote, so the server that is trusted more is given more shadows, representing more 'votes'. Since servers with more votes have more 'say' in the outcome of Gong's protocol, it raises the user's confidence in the result.

The weighting approach is also taken by Chimæra but its semantics is less clear. We refer to Figure 4.4. In the model, each certification path is given a number representing one CA's trust in another, for example, B's trust in C has degree 2. However, the value is asserted by B based on B's policy, but then shared among all CAs in the network so that the graph in Figure 4.4 can be built. The main problem with this is in the use of global trust values that were generated locally possibly through implementation of disparate local policies. The semantics problem associated with this was discussed in §4.3.1

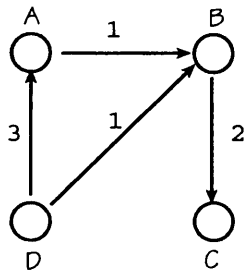


Figure 4.4: Certification paths and trust degrees in Chimæra.

Continuous trust levels

BBK, IST, Jøsang, Marsh, Poblano, NeuroGrid, Maurer-P and PolicyMaker supports continuous levels of trust, or trust levels represented as real numbers. These real numbers are modeled as representing either objective or subjective probability values. We take the view that objective probabilities represent purely syntactic forms of trust values, e.g. the beliefs of the agent does not influence the value, and subjective probabilities are intuitive 'likelihood' measurements given by the agent depending on his current beliefs. Another description of their difference is given by Krause and Clark in [KC93] as follows (where objectivists and subjectivists are practitioners of objective and subjective probabilities respectively):

Objectivists ... take the view that probability is about events we can count. They use a frequentistic definition of the probability of an event being the proportion of favourable events out of all possible events. ... The subjective view, on the other hand, is that probability is a logic of degrees of belief. The probability of an hypothesis ... is a measure of an, admittedly idealised, person's degree of belief in that hypothesis given the available evidence ..."

IST uses an **objective** representation for trust values. Trust in an agent α is a pair of real numbers between 0..1, exclusive that represents the statistical *confidence interval* for a given probability of the agent's prediction of something to be true in the future. According to the authors' definition, "trust, or reliability of agents can be conveniently expressed as a (range) probability, which will indicate the level of confidence we can have on an agent". The trust value pair is derived from the confidence interval formula that can be found in standard statistical sampling method texts such as [Coc97].

In IST's relational databse model, for each tuple in a relation there are associated contributing agents for that tuple. Each tuple can be a fact after being observed to be true, a hoax after being observed to be false or a prediction if not yet observed to be true or false. One can imagine the application of the stock market to take advantage of this classification of tuples, where a predicted share price for a specific time is then marked as true or false after being observed once that time has passed.

By taking the union of the sets of predictions (P), facts (F) and hoaxes (H) as the population of the database (D) and the union $F \cup H$ as a sample on the population D, Jamil and Sadri calculates the trust in an agent a using the following standard formula[Coc97] for calculating sampling confidence intervals.

$$\frac{|F|}{|D|} \pm z_c \sigma \sqrt{\frac{(|D| - |F \cup H|)}{|D| - 1}} \tag{4.2}$$

In the formula above, z_c is the 'Z score' for a desired confidence level c (e.g. 95%), whose value can be obtained from standard Z score tables. σ is the population standard deviation, which can be substituted with the sample standard deviation for samples greater than 30. Furthermore, a normal distribution of the population is assumed.

The authors acknowledge that this is just one way of calculating trust of an agent in their model and that a more satisfactory algorithm may be selected for the appropriate application. However, they claim that this does not affect their system as the result of the trust calculation presented here is independent of the reliability calculation of tuples in the database, which is the main concern in their work.

BBK is another model that represents trust values as objective probabilities. In BBK, two types of trust relationships are present, that is direct and recommender trusts. For both types, the trust value is a real number 0..1, derived from a function based on the number of positive and negative experiences the truster had with the trustee. Details of the function and how a trust value is derived are discussed in §4.6. For direct trust, the trust value is "an estimation of the probability that Q behaves well when being trusted" [BBK94]. For the recommender trust however, a trust value is regarded as a "degree

of similarity" between the truster and the trustee. This value is based on experiences with the agents that the recommender recommended. Although BBK's trust value semantics is one of objective probability, the function from which the value is derived does contain a subjective element. BBK's trust value functions are of the form

$$trust = 1 - \alpha^p \tag{4.3}$$

where, for direct trust, p is the number of positive experiences, and for recommender trust, p is the difference between positive and negative experiences. BBK explains that this formula actually gives the probability that the trustee has a *reliability* of more than α, and goes further to warn that α values should be high in order to make the estimations "safe". Therefore, although it is an objective metric, it contains a subjective parameter in its calculations. This makes the value meaningful only to the agent who instantiates the value of the subjective parameter α.

NeuroGrid's objective probability representation of trust values, or 'confidence values' as its author chose to call it, is based on two metrics. When a keyword search in NeuroGrid is successful the searcher will be presented with a data reference (filename or URL) that points to data that is likely to match the keyword given. For each data reference returned NeuroGrid records 1) the number of times it was recommended, r, and 2) the number of times the recommendation resulted in the data reference being accessed, a, (e.g. by clicking on it), both metrics recorded with respect to the keyword(s) used.

With the two given metrics NeuroGrid then calculates the 'confidence value' of either/both the URL and/or recommending agent, with respect to the keyword used, using a probabilistic formula. By assuming that there exists some probability p that the resulting URL will be clicked on for example, the calculated 'confidence value' tells us how close the ratio $\frac{a}{r}$ is to p. This is called the *Hoeffding Bound* and is further explained in §4.6.1.

In Maurer-P, Marsh and Jøsang, the **subjective probability** semantics is used for their trust values. In Maurer-P, statements that were defined in Maurer-D are given confidence values. So, for example, the statement expressing trust in Maurer-D, $Trust_{A,X,i}$, now is given a confidence value, represented as $conf(Trust_{A,X,i})$, in the range [0..1]. To the agent A, $conf(Trust_{A,X,i})$ is what A believes is the subjective probability of the possible statement $Trust_{A,X,i}$ being true. In Maurer's model, whether a statement can be derived from the initial assumptions of an agent, i.e. the *initial view* of an agent in Maurer's vocabulary, depends on the application of a given set of rules on the statements in the initial view. The probabilistic model extends this by first allowing the agent to attach confidence values to statements in its initial view. The confidence value of derivable statements are acquired through the prescribed formulae in Maurer-P. No further explanation was given with respect to the semantics of values given to statements in the initial view.

In Marsh, trust is given a value in the range [-1..+1] for all three types of trust defined in the model (Basic, General and Situational). These values are intuitive, as in Maurer-P. However, the semantics of three extreme values in the range was given, as shown in Table 4.10. Other values of trust lie between these extreme values. The intermediate values are asymptotic towards the extreme val-

ues because agents that are given the boundary values are assumed have no other agents more (un)trustworthy than them. Thus, as cautioned by Marsh in [Mar94a], care must be taken when using these extreme values.

Value	Semantics
-1	Complete distrust. This indicates that there is no agent less trustworthy than this.
0	Ignorance, i.e. neither trust or distrust.
+1	Blind trust, where complete trust regardless of evidence.

Table 4.10: Marsh's trust value semantics.

The third model that supports subjective probability values for trust is Jøsang. Here, trust is considered to be an instance of belief. Thus, the logic presented in Jøsang is a logic of belief, but which can be applied to trust. As briefly mentioned in §4.4, the model assumes that an opinion may constitute elements of belief, disbelief and ignorance concurrently. To represent 'opinion' numerically, a triple value $\{b, d, i\}$ was defined, where each subcomponent is a real number in the range [0..1] representing belief, disbelief and ignorance respectively. Formally, the definition, as given in [Jøs99], is as follows:

$$b + d + i = 1, \quad \{b, d, i\} \in [0, 1]^3$$

As in most belief calculi, and the models of Maurer and Marsh above, the values are intuitively assigned. To visualise the conceptual opinion triple, Jøsang provides the opinion triangle diagram, reproduced below:

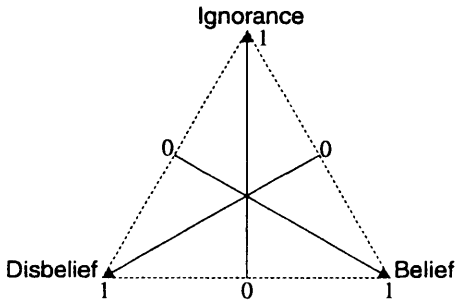


Figure 4.5: Jøsang's opinion triangle.

To illustrate the semantics of the opinion triple, the following example was given in [Jøs97a]:

> *The difference between 0.5,0.5,0.0 and 0,0,1 is that in the first case there are equally strong reasons to believe that the proposition is true as false, whereas in the second case there is no reason to believe either.*

However, when the value for ignorance is 0 (zero), the model is equivalent to a traditional probability model, with belief and disbelief values only (one being the complement of the other). The triple can also be reduced to a single *probability expectancy* value, by the following formula, as given in [Jøs99][4]:

[4]For consistency, the third component i as used in Jøsang's earlier work is used here, instead of u as used in his later papers, e.g. [Jøs99]

$E(b,d,i) = \frac{b+i}{b+d+2i}$

4.6 Trust Dynamics

In §2.7, we saw how a trust relationship is dynamic and non-monotonic. For this section, we are interested in two particular aspects of the dynamics of computer trust models. They are:

1. The initial trust *formation* phase, that is, how we get from a state of ignorance about another agent, to trusting, or distrusting, it.
2. How trust is *revised*, during the lifetime of the trust relationship, where we will look at how the level of trust is increased or decreased in light of new experience or information.

Table 4.11 provides a bird's-eye view of the models surveyed and their incorporation of these two phases of a trust relationship.

Key	Manual	Automated	Revision
BBK	•	•	•
Chimæra	•	•	
Free Haven		•	•
Freenet		•	•
Gong	•		
IST	•	•	•
Marsh		•	•
Maurer	•	•	
NeuroGrid		•	•
PGP (key trust)	•	•	
PGP (introducer trust)	•	•	
Poblano		•	•
PolicyMaker	•	•	
Rangan	•	•	
RJ		•	•
Sierra	•	•	
SPKI	•	•	
X.509	•	•	
Jøsang	•	•	

Table 4.11: Support for phases of a trust relationship.

4.6.1 Formation

The trust formation phase of social trust is characterised by the leap from ignorance to the belief in another's (un)trustworthiness, taken with caution and within a certain time-span. This phase, however, is absent from all of the models surveyed. Instead, there is the general approach of starting with an initial trust decision, followed by the establishment of a trust relationship whose treatment is not discriminated from older relationships. Although this makes for simpler trust models, it does not mirror how trust relationships exists socially: new trust relationships are very fragile while trusters are more forgiving in older relationships that usually have special 'bonds' nurtured between the two parties. Thus, in this subsection, we will only be able to review how the initial trusting decision is made.

Trust relationships can be built either by manual assertion of trust statements by the user, or automatically from existing statements, using a fixed algorithm. The models surveyed support one or both of these trust formation methods. Additionally, some models incorporate the use of a 'trusted third party' to allow formation through introduction from a third party, one that the truster trusts to recommend or introduce other agents.

The logics for analysing cryptographic protocols surveyed, that is BAN, GNY and SvO, are concerned only with static analysis of beliefs in cryptographic protocols. These logics analyse a single instance of a protocol run, and therefore are ill-suited for analysing the dynamics of trust, which spans multiple runs, or interactions with the principals in the protocol. However, their approach contributes to making the logic simpler, which is one of their goals. Due to this single-run only property, BAN, GNY and SvO are ignored in this subsection.

Gong provides only manual trust formation. In Gong, the user specifies the threshold values and manually assign 'votes' to trusted servers. The user also chooses the agents that will participate in the protocol. The values of these parameters remain constant until the user himself reassigns them. Therefore, trust formation in Gong only happens in the human user, and not supported by the model.

The majority of the models support both manual and automatic trust formation. Which method is used is determined by the type of trust involved. Typically, automatic trust formation is carried out when transitive trust is assumed, e.g. when resolving trust chains or delegated authority. X.509, SPKI, PGP, Chimæra, BBK, Maurer, PolicyMaker and Rangan all support both manual and automatic trust formation. As with Gong, the manual trust formation procedures in these models involves the user making specific assertions on trust relationships, albeit using different mechanisms and syntaxes. As an example, in PGP, the user specifically enters the trust level of an introducer, using the supplied software. Future processing of introduced certificates will then take this level of trust into account. The other models above follows the same trend in manual trust assertions. We look in more detail now at the different algorithms for automatic trust formation.

Blind Trust Formation

Chimæra, Maurer, Rangan and Jøsang allow trust relationships to be formed by blind trust chain resolution. We use the phrase 'blind resolution' to indicate that no other conditional variables or constraints are consulted before accepting the recommendation of the trusted recommender. For example, in Chimæra, trust values generated by other CAs are used at face value in calculating the value of a given trust chain (which, in Chimæra's case, is just the sum of all values in the chain). In Maurer-D, trust in another agent is formed as soon as a 'trust derivation' rule is satisfied. The general statement of the rule is that, for all $i \geq 1$, Alice trusts Cathy level i, if Bob, whom Alice trusts with level $i+1$, makes a recommendation. When $i = 0$, the trust relationship is a direct one. The formal description of the rule is given in 4.4 below, where $Aut_{Alice,Bob}$ is the statement for Alice's belief in the authenticity of Bob's public key, $Trust_{A,Bob,i}$ is the statement indicating Alice's trust in Bob for level i, and $Rec_{Bob,Cathy,i}$ indicates the recommendation from Bob about Cathy with respect to trust in Cathy of level i. In a similar manner, Rangan and Jøsang relies on transitivity to acquire new beliefs. Likewise for Poblano and Sierra to form relationships with nodes in a trust chain.

$$\forall Bob, Cathy, i \geq 1 : Aut_{Alice,Bob}, Trust_{Alice,Bob,i+1}, Rec_{Bob,Cathy,i} \vdash Trust_{Alice,Cathy,i} \quad (4.4)$$

In Free Haven, new nodes in the network first make contact with 'introducer' nodes that assigns a value of zero as the new node's trust value. The introducer then forward an 'introduction referral' to other nodes in the system so they may be aware of this new node and start transacting with it. Although any node in Free Haven can be an introducer, the model does not specify how trust in the introducer itself can affect the introducer's introductions.

Further details of the algorithms of Jøsang and Poblano are looked at when we discuss trust value formation below.

The problem with building trust blindly based on the recommendations of others is due to the subjective nature of trust. Thus, to assume that all agents cognitively process trust in the same way and then to go on and define a universal fixed trust algorithm is not a reasonable approach. Furthermore, trust is not necessarily transitive, as Jøsang himself mentioned [Jøs96, Jøs99]. What is required is the flexibility to allow agents control over the trust decision making process. This is where constraints come in.

Formation with Constraints

X.509, BBK, PolicyMaker, NeuroGrid and Sierra support constraints on when and how trust chain resolution may succeed, with the latter two using only minimal constraints compared with the previous three. Additionally, these models are also flexible enough such that 'blind resolution' is supported, by ignoring the constraints, should the user want this option. Each of these models support constraints in different ways. However, before going into more detail about the constraint mechanism in these models, we will briefly examine one other model, i.e. SPKI, whose support for constraints sits somewhere between those of the blind resolution approach and full constraint support.

In SPKI, trust is not resolved blindly. Alice, for instance, will trust Bob, with respect to a certain authorisation identifier X, if there is a valid authorisation chain between Alice and Bob. A valid authorisation chain here means that the intermediate entities have been delegated authorisation X from Alice, directly or transitively, and also allowed to delegate X (or a subset of X) to other entities. For example (see Figure 4.6 below), Alice granted Cathy authorisation X and the ability to delegate X. Cathy then delegates the same to David. David grants X to Bob, but not the ability to delegate X. This is a valid authorisation chain.

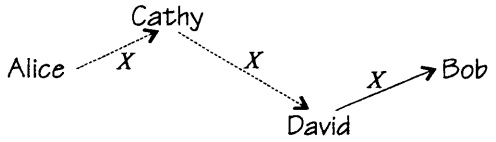


Figure 4.6: A valid SPKI authorisation chain. Dotted arrow indicates delegation.

Thus, even though trust is not resolved blindly, there is no support for imposing sophisticated constraints on the trust relationships in SPKI, other than the existing implicit constraints, based on valid

authorisation chains.

NeuroGrid only allows constraints on the length of a valid trust chain, also called the Time-to-live (TTL) constraint. The TTL is specified each time a search request is generated, so TTLs can be set on a case-by-case basis. Sierra also uses chain lengths as a constraining factor on trust relationships. However, Sierra uses a 'Node Distance Distrust Table' to specify how much trust to grant an edge in a trust chain. The further away the edge is from the origin, the lower the trust value, until it stops when an edge of degree n is assigned a trust value of zero. An example is shown in Figure 4.7.

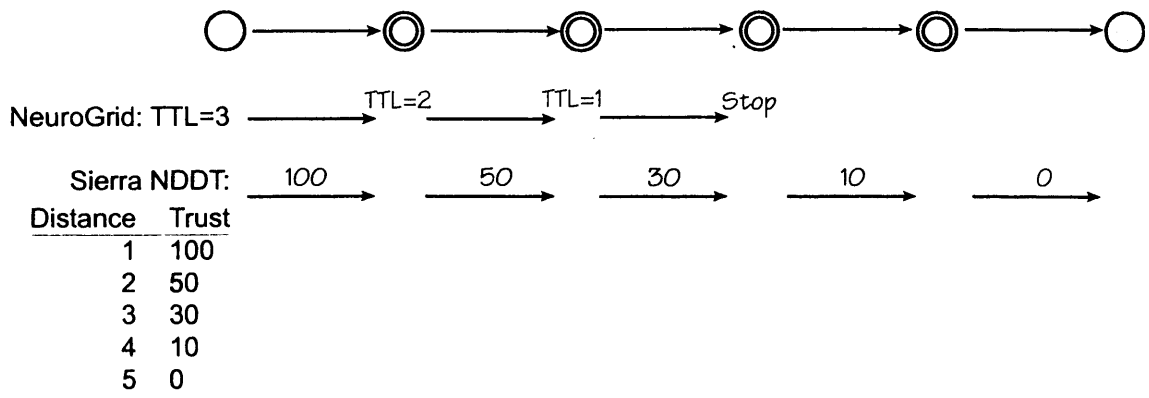


Figure 4.7: Constraining trust relationship formation using trust chain length (NeuroGrid's TTL) or by weighing paths based on distance from source (Sierra's Node Distance Distrust Table).

X.509 allows a lot more flexibility, by allowing users to define arbitrary constraints by way of certificate extensions, which were mentioned briefly in §4.3.5. Although users are free to define their own extensions, X.509 recommends a *Certification Path Constraint* extension to handle trust chain processing. The constraints contained in this extension are generated by CAs and attached to other CA-certificates, i.e. this extension is for a CA to impose constraints on certificates it issues for other CAs. Three types of constraints are recommended by X.509:

Basic This constraint says whether the subject in this certificate may be a CA to other subjects. It also allows a certification path length constraint to be imposed – the number of certificates in the chain starting from this certificate must not exceed this number.

Name If this is a CA certificate, then the subject of all subsequent certificates in the chain must exist within this given name space. This constraint only works with a hierarchical naming structure.

Policy This says that, from a certain nominated point onwards in the trust chain, all certificates must carry an acceptable policy identifier that indicates the policy of the CA identified as the subject of that certificate.

In a simple example (shown in Figure 4.8), Alice wants to verify the authenticity of a public key that is claimed to belong to Bob, whose email address is `bob@picosoft.com`. Bob's certificate is signed by CA2, whose certificate is signed by CA1. Furthermore, CA1 specified a *Name* constraint on CA2's certificate, trusting it to only certify within the `soleil.com` namespace. Therefore, when Alice unwraps Bob's certificate, she discovers the Name constraint and finds that `bob@picosoft.com` is outside of the `soleil.com` namespace. She then verifies that CA2's

certificate was signed by CA1. Now, whether Alice trusts CA1 or not, this authentication run will fail, because of the constraint. If Alice does not trust CA1 with regards to imposing the Name constraint on CA2, then Alice will have to manually find other means to verify the authenticity of Bob's public key. In X.509, to resolve a trust chain, there must first be a trusted public key which will be used in verifying the first certificate in the chain. In this example, CA1 is Alice's trusted public key.

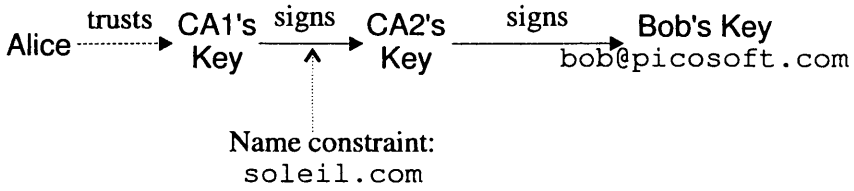


Figure 4.8: Example of an X.509 certification chain with *Name* constraint.

Even greater flexibility is allowed in BBK. Agents are able form arbitrary predicates that define acceptable or non-acceptable entities in a trust chain. However, these predicates are only used in two constraint classes; the *when.path* and *when.target* constraint classes. The former class filters intermediary agents in the chain (or path), and the latter class filters the end-entity in the chain. The following statement in BBK, for example, says that Alice trusts Bob with respect to giving her a recommendation about X, with one or more intermediary agents in the trust chain (also called the recommendation path), as long as the intermediary agents are aged over 25 years old, and the target (end) agent is earning over $12,000 a year. This could perhaps be an instance of a credit status check by Alice on the target entity.

$$\textit{Alice trusts.rec}_X \textit{ Bob when.path } [\textit{age} > 25] \textit{ when.target } [\textit{salary} > 12,000] \tag{4.5}$$

PolicyMaker's free-form approach to policy specification provides the most flexible method of defining constraints on trust relationships. All policies must be written in the PolicyMaker language. However, PolicyMaker does not specify specific constructs for handling constraints per se. What is available is a method of describing the conditions for which an action, requested by an agent represented by his public-key, is allowed. An example policy, taken from [BFL96] is given below. This example policy binds Alice's PGP key, referenced with her key fingerprint in the assertion, to the ability to sign messages containing the appropriate values in the From and Organization fields in the email header. In other words, the policy accepts as valid emails signed with the given key if the email is from the key's owner.

```
policy ASSERTS
       pgp: "0xf001223hjasj887498jh8467qwyu98f"
       WHERE PREDICATE = regexp:"(From:Alice) &&
                         (Organization: Bob Labs)";
```

Figure 4.9: Example policy in PolicyMaker.

Forming Trust Values

In addition to forming trust relationships, some models also include algorithms for attaching values or levels of trust to the initially formed trust relationships. Sierra, NeuroGrid, Poblano, Free Haven, PGP, BBK, Chimæra, IST, Jøsang, Maurer-P and Marsh take this approach. The semantics for their values were discussed in §4.5.2 earlier. Apart from Chimæra, whose values are attached to trust relationships in an ad hoc manner, these models provide algorithms for calculating trust values when forming new trust relationships. The algorithms can be subdivided into two categories. The first category uses knowledge from past experiences and/or trust dispositions to form new trust relationships. Algorithms in this category are described under the heading "Formation with Experience and Disposition" below. The second category contains fixed algorithms which calculate trust values in a static manner. BBK's Recommendation Trust inference rule and Jøsang's recommendation algorithm are in this category.

Sierra's main design goal is to provide a framework for propagating trust values and for measuring such propagation paths. It leaves the final trust value calculation to an external 'plug-in' module that is open for third-party implementation. As such it does not define how or whether trust should be calculated from experience. Thus this subsection will not contain further discussions on Sierra.

BBK's rule of inference for deriving a Recommendation Trust value is given in [BBK94] for forming new trust relationships with another agent whom we may trust for providing recommendations with respect to a certain context or trust category X. If, for example, Alice trusts Bob as a recommender with value v_1 and Bob recommends Cathy as a recommender with value v_2, then the Recommender Trust value of Cathy is given by Equation (4.6) below, which is simply a product of the trust values. The multiplication approach may seem intuitive as it serves to ensure that "the trust value descends as the recommendation path grows", as the authors put it, but no other reason was given as to why multiplication was preferred to other operations which can achieve the same desired effect.

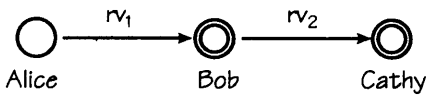


Figure 4.10: Alice trusts Bob as recommender who trusts Cathy as recommender.

$$trust = rv_1.rv_2 \tag{4.6}$$

If Bob's trust in Cathy was a Direct Trust relationship, then the trust value is derived using Equation (4.7) below:

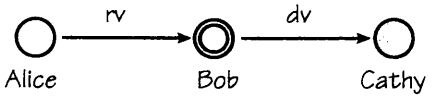


Figure 4.11: Alice trusts Bob as recommender who trusts Cathy directly for some action.

$$trust = 1 - (1 - dv)^{rv} \tag{4.7}$$

Poblano resolves the trust chain for direct trust in a different manner, using the averaging approach instead. Given the chain between Alice and Danny below, the equation for direct trust is gien in Equation (4.8), where n is the number of intermediate recommenders.

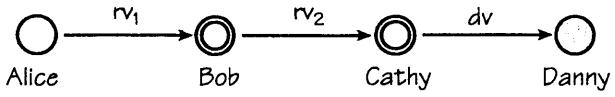


Figure 4.12: Alice trusts Bob as recommender who trusts Cathy as recommender who trusts Danny directly for some action.

$$\frac{1}{4n}(\sum_{i=1}^{n} rv_i) \times dv \tag{4.8}$$

Jøsang too provides operations for forming trust values. We recall from §4.5.2 that in Jøsang, a trust value, or more accurately an agent's belief in a statement s, is represented by an opinion triple containing belief, disbelief and ignorance components, as shown formally below.

$$belief = b + d + i = 1, \quad \{b, d, i\} \in [0, 1]^3$$

Jøsang provides an operation for resolving recommendations, a situation similar to BBK's recommendation trust resolution trust above. However, in Jøsang's case, the formula allows Alice to form an opinion about an arbitrary statement s, based on Bob's opinion of s, which is a more general case of the BBK scenario above where values are formed and attached directly to the recommended agent. Thus, if Bob gives his opinion $\omega_B^A = \{b_B^A, d_B^A, i_B^A\}$ about a statement s to Alice, and Alice's opinion of Bob's recommendations is $\omega_s^B = \{b_s^B, d_s^B, i_s^B\}$, then Alice's opinion of the statement s can be resolved by the following (Alice and Bob are shortened to A and B respectively):

$$\omega_s^{AB} = \omega_B^A \otimes \omega_s^B = \{b_s^{AB}, d_s^{AB}, i_s^{AB}\} \tag{4.9}$$

where

$$\begin{aligned} b_s^{AB} &= b_B^A b_s^B, \\ d_s^{AB} &= d_B^A d_s^B, \\ u_s^{AB} &= d_B^A + u_B^A + b_B^A u_s^B. \end{aligned}$$

In PGP, the algorithm for calculating the trust value for a key is given in Figure 4.13. Although not strictly dealing with a form of 'trust', this algorithm of PGP's attaches a value to a 'belief' in

the validity of a key, which is akin to Jøsang's approach of attaching values to an agent's belief in a particular statement. Since trust is also a belief, we think this is worth highlighting here. In summary, what the algorithm does is to first sum the number of partially trusted and completely trusted signatures on the key. The sums are then compared with PGP's COMPLETES_NEEDED and MARGINALS_NEEDED parameters, or its Dispositional Trust variables, as we discussed in §4.5.2. If either of these parameters are met, then the key is considered to be 'completely valid', otherwise, it is 'marginally valid'.

```
Phase 1: Scan signatures on key
FOR each signature
    IF signature is completely valid
        IF key trust IN {undefined,unknown,untrusted}
            ignore signature
        ELSIF key trust is marginal
            accumulate marginals_counter
        ELSIF key trust is complete
            accumulate completes_counter
        ENDIF
    ELSE
        ignore signature
    ENDIF
ENDFOR

Phase 2: Verify key validity
IF (marginals_counter > 0) or (completes_counter > 0)
    IF (marginals_counter >= MARGINALS_NEEDED) or
    (completes_counter >= COMPLETES_NEEDED)
        mark key validity as 'complete'
    ELSE
        mark key validity as 'marginal'
    ENDIF
ELSE
    mark key validity as 'unknown'
ENDIF
```

Figure 4.13: Calculating key validity level in PGP.

Another example where new trust relationships are given values is in Marsh's Situational Trust formulæ[Mar94a]. In this model, two scenarios are identified and discussed, i.e.

1. When the target agent is not known at all, and
2. When the target agent is known, but not in the currently considered situation (or that there had been no prior experience with that agent for the current situation).

In these scenarios, Marsh bases the new trust values on the truster's Basic Trust value (in scenario 1) or General Trust value (in scenario 2). Marsh's Basic Trust represent the general trusting disposition of an agent, based on its previous experiences with all agents in all situations. General Trust is the truster's level of trust in a specific agent, regardless of any particular situation. In the formalism, the new trust values for scenarios 1 and 2 above are simply the truster's Basic (scenario 1) or General (scenario 2) Trust values multiplied by two other variables representing the truster's perception of

the Utility and Importance of the situation in question. Note that trust relationships in Marsh are highly dynamic and only lasts for the duration of the situation involved. Future interactions will require a recalculation of the trust values, based on various factors like the utility and importance of the situation. These operations are shown more formally below, where Alice is the truster, Bob the target, α is the situation in question, I is the importance value and U is the utility value. Thus, for scenario 1, the new Situational Trust value $T_{Alice}(Bob, \alpha)$ is

$$T_{Alice}(Bob, \alpha) = T_{Alice} \times U_{Alice}(\alpha) \times I_{Alice}(\alpha) \tag{4.10}$$

and for scenario 2, the new value is given by

$$T_{Alice}(Bob, \alpha) = T_{Alice}(Bob) \times U_{Alice}(\alpha) \times I_{Alice}(\alpha) \tag{4.11}$$

In IST, trust values are based purely on statistical information about an agent's past behaviour. The method prescribed by IST in calculating trust values is based purely on statistical estimation and takes only positive and negative experiences with an agent into account. Since new agents do not yet have a 'history' with the truster, all new trust 'relationships' in IST have a trust value of 0 (zero). In effect, this model does not give an agent a trust value until there has been at least one experience with it. This gives the model a very short 'tentative' phase, akin to the 'trust formation phase' in social trust (see §2.7.1), albeit lasting only until the first experience. Unlike most models, however, IST does not purge 'untrustworthy' agents from the system once they behave negatively. Agents are continually monitored for both positive and negative experiences, which allows their behaviour to be estimated from statistical data over time. Thus, an agent can continue to behave badly and remain in the system.

It has to be said, however, that the nature of IST's application domain allows such long-term residence of rogue agents - the agents are merely providing input into a database. The system then merely calculates the statistics based on the inputs and observed facts by the human user and then presents the results back to the user for her make her own decision. Such applications make the somewhat lenient approach taken by IST suitable because the results serve only as a guide to decision making. It is a different matter in a system where agents have the opportunity to be directly malicious and cause irreparable damage. In such cases, more care will have to be taken in determining the trustworthiness of agents.

We briefly looked at NeuroGrid's trust value representation on page 99. Two metrics are recorded for each search result and they are 1) the number of times it was recommended, r, and 2) the number of times it was accessed, a, (e.g. by clicking on it), both metrics recorded with respect to the keyword(s) used. By assuming that there exists a fixed and unknown probability p that the returned results will be accessed, a click on a URL for example, $\frac{a}{r}$ can be thought of as an estimate of p. A fundamental principle in statistics called the law of large numbers states that the larger the value of r the closer $\frac{a}{r}$ is to p. It would be more useful however to know *how* close $\frac{a}{r}$ is to p. For this NeuroGrid uses the

Hoeffding Bound that states that gien a fixed error ϵ, the probability that a/b is different from p by at least ϵ is:

$$Pr[|\ (a/b) - p\ | > \epsilon] \leq 2\epsilon^{-2b\epsilon^2}$$

So to find out the minimum probability that $\frac{a}{r}$ deviates no more than ϵ from p, we will do the following:

$$\begin{aligned} Pr[|\ \frac{a}{r} - p\ | > \epsilon] &= Pr[(\frac{a}{r} > \epsilon) \quad OR \quad (p - \frac{a}{r} > \epsilon)] \\ &\leq Pr[\frac{a}{r} > \epsilon] + Pr[p - \frac{a}{r} > \epsilon] \\ &\leq 2\epsilon^{-2r\epsilon^2} \end{aligned}$$

So if we have $r = 1000$, $a = 500$ and $\epsilon = 0.05$, then the probability that $\frac{a}{r} - p$ is less than 0.05 is 98.7%. If we also have another recommender with $r = 500$ and $a = 250$, this will give is a probability of 83.5% for the same error margin. Therefore, even though the value of $\frac{a}{r}$ is the same for both recommenders, i.e. 0.5, the fact that the first recommender has higher frequencies of recommendation and access means the confidence factor in him is higher.

It has been pointed out by one of the advisors to this work that the Neurogrid model exhibit properties of ergodic dynamic systems[Sha05]. The ergodic theorem is similar to the law of large numbers, i.e. that a sufficiently large sample is representative of the population itself.

Formation with Experience and Disposition

As Jøsang said in [Jøs96], the right kind of trust model for distributed systems is one which is based as much as possible on knowledge. Our discussions on social trust also showed us that knowledge and its role in forming trust dispositions is important for forming trust opinions about agents we have never met before. The use of knowledge and disposition was demonstrated in PGP as we saw above, among other models. How disposition and knowledge are used, however, differs from one model to another. In general, they were used in one of two ways: based on the (human) user's own disposition, or based on a generalised history of past experiences.

PGP is an example of the first method. The model uses dispositional parameters expressed as the number of trustworthy agents required in a consensus about the target agent before the target agent is deemed trustworthy. This parameter can only be adjusted manually by the user and applies to all trusted agents in his database. This method is rather rigid and dynamically coarse, as in most cases users work with a small number of recommender agents and only occasionally make adjustments to the dispositional parameter (if at all). Furthermore, not being able to express different dispositional parameters for different agents or groups of agents forces the user to make general assertions for all agents that she trusts. Marsh has a dispositional trust value variable which is used for forming new trust relationships. However, like PGP, this is also a variable that the user must adjust manually.

The second method is supported by BBK, where past experiences are recorded as numbers of positive

or negative experiences. However, this does not shape an agent's general disposition which can be applied to forming new direct trust relationships. Instead, it is only used, as far as trust formation is concerned, as a generalised similarity measure of a recommender's past recommendations and the agent's own experiences with the recommended agent. This is then used as knowledge to derive trust values for any future recommendations from that recommender. As an example, suppose Alice trusts Bob's recommendation with value v_1 and Bob gives a recommendation of v_2 for Cathy (as shown in the diagram below). The Direct Trust value for Alice's trust in Cathy is calculated using Equation (4.12) below

$$v_1 \odot v_2 = 1 - (1 - v_2)^{v_1} \tag{4.12}$$

As we have encountered earlier, v_1 is the recommendation trust value, which is calculated based on positive (p) and negative (n) experiences as shown in Equation (4.3). According to the authors, the above formula is a "result of the computation of the direct trust values", as given by Equation (4.12), "and the semantics of the recommendation trust values". In other words, in the direct trust equation Equation (4.12), $1 - \alpha^i$, α was replaced by $1 - v_2$ and i by v_1 to produce Equation (4.12) above.

NeuroGrid and Poblano also uses past experiences to form trust values in a similar manner, using counts of positive or negative experiences in statistical measures.

Combining Recommendations or Trust Chains

In order to form a trust opinion or belief based on more than one recommendations, the recommendations received must be combined in some way. BBK, Jøsang, PGP and Maurer-P describe algorithms for combining recommendations in their models. Since we cannot be sure that these recommendations come from completely independent recommendation paths, it is also necessary to consider path independence when combining values. This can be visualised better if we treat these paths as edges in a graph. The example graph in Figure 4.14 shows recommendation (broken line) and direct (solid line) trust relationships between various agents, which Alice can visualise after receiving recommendations from her trusted recommenders. We can see that although Alice sees three recommendations rec_1, rec_2 and rec_3, the first two have in fact, at one point, emanated from a single source, i.e. Eric. In other words, the paths for recommendations rec_1 and rec_2 intersected at Eric and are therefore not independent of one another. Examples of how this problem is handled in the models surveyed are discussed below.

The combination of recommendations is closely related to the semantics of a recommendation and the specifics of how the model incorporates values. Therefore, it is not a practical exercise to find a general model for combining recommendations due to this implementation specific dependency. However, we will show examples from PGP, BBK and Jøsang below as examples on how they are being applied in practice.

In PGP, recommendations are represented by signatures on the recommended public key. The details of combining these recommendations have been explained and shown in Figure 4.13 above. For completion we will just mention here that in PGP, the combination of recommendations constitute

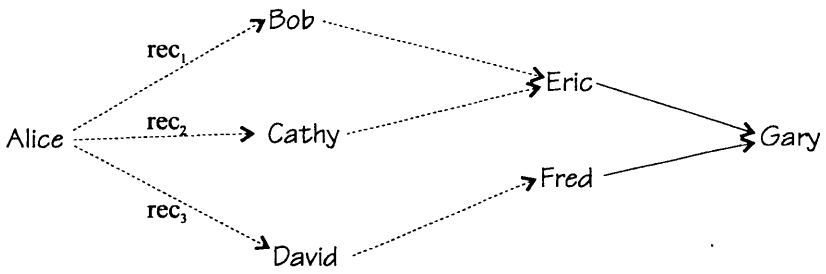


Figure 4.14: Example recommendation graph. Alice receives recommendations about Gary from Bob, Cathy and David, who received those recommendations from Eric (Bob and Cathy) and Fred (David).

accumulating the number of marginally and fully trusted signatures and then assigning the final validity value on the public key based on the user's preset conditions of how many fully or marginally trusted signatures are required before a public key is considered valid.

In BBK, several recommendation trust relationships may exist between two entities, as shown in Figure 4.15 below. This situation may arise, for example, after receiving several recommendations about a particular agent's trustworthiness with respect to recommending other agents. To combine these *recommendation trust* paths into one, the arithmetic mean of the trust values of each path is taken to be the new single value. This is formally written in Equation (4.13), where v_i represent a recommended trust value. BBK justifies the requirement of using the arithmetic mean by the requirement that no single value should influence the resulting combined value. However, it is still possible for a single value to be influential to a certain degree, compounded by the fact that interpretations of the values are subjective. For example, with four recommendations of 0.01, 0.01, 0.01 and 0.99, the combined trust value is 0.255, which may not have any logical relation to the four sample of values used. An additional interesting comment on BBK's method is made by Jøsang in [Jøs97c], where he said that the decision not to use either extreme of the set of recommended values to combine indicates a distrust in the model itself. In other words, as Jøsang puts it, there are uncertainties about uncertainties in the system, or higher order uncertainties exist in the model.

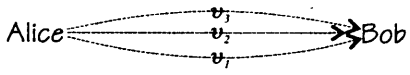


Figure 4.15: Several recommendation trust relationships may exist between two entities in BBK.

$$\text{combined recommendation trust} = \frac{1}{n}\sum_{i=1}^{n} v_i \tag{4.13}$$

To combine several *direct trust* relationships in BBK, the new direct trust relationships are first calculated for each recommendation. They are then grouped according to their common last recommending entities. For example, in the example in Figure 4.14 above, direct trust relationships resulting from rec_1 and rec_2 will be grouped together because they share a common last recommending entity, that is Eric. Values for members of these groups are first combined by the following formula for n number of paths, where v_i are their trust values:

$$\text{combined direct trust} = \sqrt[n]{\prod_{i=1}^{n}(1 - v_i)} \tag{4.14}$$

Once this is resolved, the remaining values, which are now supposed to be independent of each other, are multiplied together and taken away from 1 to get the final combined trust value. The complete formula is given in Equation (4.15) below, where g represents the group containing paths with common last recommending entities, m represent the number of those groups, n represent the number of direct trust relationships in each group and $v_{i,g}$ represent their values.

$$v_{combined} = 1 - \prod_{g=1}^{m} \sqrt[n]{\prod_{i=1}^{n}(1 - v_{i,g})} \tag{4.15}$$

Jøsang specifies a consensus rule for combining opinions of two agents, Alice (A) and Bob (B), to obtain a single opinion value about a statement s. One way to visualise this operation is to imagine that the consensus opinion is held by a single imaginary agent [Alice,Bob] who holds the combined opinion of both. This operation is shown as:

$$\omega_s^{AB} = \omega_B^A \oplus \omega_s^B = \{b_s^{AB}, d_s^{AB}, i_s^{AB}\} \tag{4.16}$$

where

$$\begin{cases} b_s^{A,B} &= (b_s^A u_s^B + b_s^B u_s^A)/(u_s^A + u_s^B + u_s^A u_s^B), \\ d_s^{A,B} &= (d_s^A u_s^B + d_s^B u_s^A)/(u_s^A + u_s^B + u_s^A u_s^B), \\ u_s^{A,B} &= (a_s^A u_s^B)/(u_s^A + u_s^B + u_s^A u_s^B). \end{cases}$$

Equation (4.16) above is valid if the opinions of Alice and Bob were obtained *independently*, i.e. if they both made observations about an event which produced their respective opinions over two different periods. Concensus between dependent and partially dependent opinions follow different rules to the above equation and are also considered in Jøsang [JK98]. Since we are interested in the general principles of how values are formed, the minor differences between these different situations will be ignored here. Furthermore, it is noted that Jøsang did not specify how this rule can be generalised to consensus between arbitrary number of opinions.

4.6.2 Revision

Beyond the trust formation phase, trust becomes less fragile. However, trust is non-monotonic and thus the level of trust in an agent can rise or fall depending on various factors, including time and experience. In §2.7.2, the importance of experience or historical information to trust was discussed.

Trust revision is in Freenet, Free Haven, Poblano, BBK, RJ, IST and Marsh. Here, we are interested in mechanisms that prescribe algorithms or methods for the revision process. Therefore, although it can be argued that the remaining models support trust revision by way of manual manipulation by the user, we will ignore them here.

RJ's simplistic revision approach only takes into account the last experience had with another agent and an agent makes its decision on future interaction based on this short-term memory. Trust revision is simply the act of not trusting, therefore not interacting, with the untrusted agent. An agent is prompted to revise, i.e. trust a new trustee, when a) its current trustee defects, or b) it hears about a 'better' agent from another agent, through word of mouth. Approach b) is safe in RJ because it models buyer and seller agents and in the model the buyers, who spread reputation information between themselves about sellers, are assumed to be completely benign - only sellers defect. The danger of malicious rumour-mongering is thus not an issue in this scenario. Furthermore, since all buyers actively seek out the 'best' (or, most reputable) seller, the system naturally weeds out 'bad' sellers. This is a very effective form of distributed social control.

A similar approach to RJ is taken by Freenet – if a node continually fails to return a requested object then eventually it will be removed from the requester's routing table and never contacted again.

In Free Haven an agent can be assigned two types of Trust values, Trust and Meta-trust. The former relates to experiences with the trustee agent's performance and the latter to the trustee's utterances. Each trust value also has an associated 'confidence' value which is an indication of how sure the truster is in giving the trustee that trust value. New agents that get introduced into the system are assigned a trust value of zero by introducers for the same reason as IST above. A small confidence value is attached to this trust value. It assumed that some kind of generic metric is used for assigning trust values based on experience as none is given in the literature. However, interpreting recommended trust values were outlined and these were given for interpreting the Meta-trust value, and both the confidence values for Trust and Meta-trust.

If Alice receives a recommendation from Bob about Cathy containing the following $[Trust = T, Confidence = C, Meta - trust = M, Meta - trustConfidence = F]$ and Alice already knows and rates Cathy as $[Trust = t, Confidence = c, Meta - trust = m, Meta - trustConfidence = f]$ then according to the Free Haven model Alice does the following:

1. Calculates new t as follows:

$$\frac{((T - t) \times M)}{c} \tag{4.17}$$

2. If t's magnitude increases then increase c, else decrease c. In other words if we have increased a positive t or decreased a negative t then increment c, giving us more confidence in t.

3. Then if the recommended ratings agree with ours, i.e. if $T = t$ and $M = m$, Alice increases her Meta-trust and Meta-trust Confidence for Bob. The rationale for this is that "if that node agrees with us, it must be worth listening to".

Unfortunately it wasn't clear what Alice should do if Cathy was unknown. Furthermore the rationale

for the trust calculation formula (4.17) was not given.

BBK relies on the concept of positive or negative experiences for making revisions to its trust value. Each positive or negative experience is independent of how much the actual gain or loss was incurred in the defection. BBK treats direct and recommendation trust revisions differently. A direct trust relationship can only exist when there has not been a single negative experience with the trustee. This is rather peculiar since this implies that a trustee is not allowed to make any mistake at all, even after a very long and fruitful relationship! What is modeled is a probability of a trustee having a certain level of reliability, α, based on the number of positive experiences had with it, p. The revision of a trustee's trust value is automatically made each time the trust value is calculated, as the variable for the number of positive experiences p is part of the trust calculation formula in Equation (4.3) shown again below.

$$trust = 1 - \alpha^p$$

BBK's recommendation trust revision takes both positive and negative experiences into account. Again, the experience counter variables are part of the trust calculation formula and thus the revised trust value is automatically obtained when a trust value is calculated. The formula for calculating recommendation trust is given below, where p and n are the number of positive and negative experiences encountered with the recommender respectively, and α is the reliability value for which the truster want to know if the trustee can meet.

$$v_r(p,n) = \begin{cases} 1 - \alpha^{(p-n)} & \text{if } p > n, \\ 0 & \text{else.} \end{cases}$$

Since the *intensity* of the positive or negative experience is not captured in BBK, there is the problem that 'minor' and 'major' experiences are treated equally. This also forces the model to assign a trust degree of zero when negative experiences equal positive ones, even when the negative experiences could have been only, in the eyes of the truster, minor errors made by the trustee. This problem can be resolved by allowing the model to handle different degrees of experience.

In Marsh, agents may revise their basic, general and situational trusts at any time using a set of prescribed formulæ. There is, however, a general rule for the frequency with which these trusts can be modified. situational trust has the highest frequency of revision, followed by general trust and lastly basic trust. The reason for this was given in [JM97] and we quote the authors here:

> *... contextual trust is modified the most as the truster can consider with certainty the most recent behaviour of the trustee in the current context. General trust is modified less than contextual trust because the interaction was in only one of perhaps several contexts. The smallest modification is made to basic trust because the interaction was with one of perhaps several individuals in one of perhaps several contexts.*

In Marsh, the notion of *groups* of agents is important as it forms the foundation of the model's trust revision metric. All agents are considered to be part of a group of agents [Mar94a]. The definition of a group's boundary is arbitrary. To modify trust values, agents must know its own dispositional

trust value, as well as the disposition of the group as a whole.

> *Members of a group are considered to have a level to which they are prepared to modify trust values. This is determined by a member's own disposition and by the disposition of the group as a whole. Here we take the view that as a basic trust increases so will the level of modification, and as basic trust decreases, so will the level of modification. [Mar94a]*

When trustee y cooperates or does not cooperate with truster x in situation α the modification values are, respectively, added to or subtracted from the trust values, as shown below.

$$\begin{aligned} T_x(y,\alpha) &= T_x(y,\alpha) \pm M_{T_x(y,\alpha)} \\ T_x(y) &= T_x(y) \pm M_{T_x(y)} \\ T_x &= T_x \pm M_{T_x} \end{aligned}$$

where,

$$M_{T_x} < M_{T_x(y)} < M_{T_x(y,\alpha)}$$

The modification values are calculated as follows:

$$\begin{aligned} M_{T_x} &= \frac{\frac{1+T_x}{2}}{\mid G_1 \mid \times \mid S_{G_1} \mid} \times M_{G_1} \\ M_{T_x(y)} &= \frac{\frac{1+T_x}{2}}{S_{G_1}} \times M_{G_1} \\ M_{T_x(y,\alpha)} &= \frac{1+T_x}{2} \times M_{G_1} \end{aligned}$$

M_{G_1} is calculated as follows:

$$M_{G_1} = \frac{1+T_{G_1}}{2}$$

where, T_{G_1}, the average basic trust of the group members, is:

$$T_{G_1} = \frac{1}{|G_1|} \sum_{\omega \in G_1} T_\omega$$

Poblano provides three separate algorithms for trust updating. The first updates the trust, or 'confidence', in the object when the truster receives a recommended trust value about that object. The new trust value is calculated as follows.

$$a.oldTrustValue + b.recommendedTrustValue + c.latestUserRating,$$
$$\text{where} \quad a + b + c = 1$$

The idea of the formula above is to weigh the three relevant metrics and then sum them to form the new value. The values of a and b depends on whether the object is more popular on the truster's or recommender's node. c is usually much larger than a or b as the user's rating of the object is the most important metric. The given formula above is rather ad hoc and it is difficult to make sense of the rationale behind it. A more reasonable approach is Poblano's second update function, for updating trust based on feedback given by a requester's rating of the object you just recommended. This update function is given in the formula below. In essence, the new trust value is the average of the old value and the fed back value after being adjusted according to our confidence in the agent that provided the feedback.

$$\frac{oldTrustValue + (feedbackTrustValue \times feedbackerTrustValue/4)}{2} \tag{4.18}$$

Finally, Poblano's third update function is about revising trust in the peer (who made a recommendation) after observing the quality of the object he recommended. It is a function of the current trust we have in the peer and all previous recommendations that the peer made, shown in the formula below. So for a given peer P,

$$\frac{oldPeerTrustValue + \frac{1}{|a|}\sum_{a \in K} trustInRecommendedObject_P^a}{2} \tag{4.19}$$

The variable $trustInRecommendedObject_P^a$ is the rating given to the object recommended by P when queried for keyword a. Thus the statement $\sum_{a \in K} trustInRecommendedObject_P^a$ in the formula above means the sum of all such *trustInRecommendedObject* for all keywords we queried P for. This gives a generalised trust in those objects which P recommended. This value is then averaged with the last trust value in P, giving the new trust value.

IST's dynamic trust calculation approach makes trust revision an inherent feature of its model. An agent's trust value is calculated based on the number of facts and hoaxes it contributed to the database, which keeps a record of all contributions made by the agent, together with indicators of whether the information contributed have been observed to be facts or hoaxes. Unobserved information is neither a fact or hoax. Details of this representation and calculation were discussed on page 100. As there are no static trust statements used in the model and values are calculated on-the-fly, there is no need to handle revision explicitly in IST - it is, as we said above, inherent in the model.

An additional item of consideration in this section is an agent's capability for remembering the past. In practical terms, more stored historical information equals more physical memory required for the agent software. Although the price of physical volatile or semi-permanent computer memory is

relatively very cheap at the time of writing, unbounded memory can present performance problems. Additionally, mobile code (e.g. mobile agents or downloaded software plugins) will have a limit on the size of its storage space.

The approaches of Freenet, Free Haven, RJ, BBK and Marsh uses single values to represent the amount of trust. This value is usually a floating point number between 0 and 1, and therefore takes up very little physical memory space. For example, if the floating point representation used takes up 4 bytes of memory, then the agent will only require about 40 Kilobytes of memory to store trust values for ten thousand trustees. Furthermore, Marsh justifies the use of single values by saying that "one of the benefits of trust is that in one value, a large amount of past experience can be presented. That is, since the trust in an agent will rise and fall according to the past experiences, the truster has had with an agent, the final value is a good approximation of how well the trustee *should* be trusted" [Mar94a].

However, single values do not allow the truster to determine whether the last experience with the trustee is consistent with its past behaviour or an anomaly. A better approximation of an agent's reputation can only be formed when there is complete information. This supports Jøsang's thesis that a trust model for distributed systems must, as far as possible, be based on knowledge. Knowledge is what IST has in abundance as it keeps a record of all information contributed by an agent and further marks the information tuples as fact or hoax. This approach is satisfactory for IST as it is a model for representing information sources for existing databases, which are designed to handle large amounts data anyway (although their approach does approximately double the amount of storage required for each tuple). For memory constrained applications, however, like embedded systems or mobile agents, this model will present problems.

What kind of information to use and how it should be represented will depend on the application of the trust model and its constraints on memory. Memory, on the other hand, impacts the strength of the trust model used. A balance, or compromise, must be made when designing an effective trust model. Thus, any general trust model must make clear the memory constraints it imposes, or be general enough such that the decision on how much memory to use is commensurate with the expected effectiveness of the model.

Additionally, the disposition of the model (or its designers) itself is evident from the way it handles trust, especially with respect to how lenient it is in tolerating 'bad behaviour' from others. As an example, BBK is a very pessimistic model because a single negative experience will sever the trust relationship forever. IST, on the other hand, will welcome any unknown agent, and will continue to host contributions from 'bad' agents - it merely reports the truth about the trustworthiness of those bad agents and it is up to the user decide what actions to take.

4.7 Chapter Summary

In this chapter, we have surveyed 22 current trust models and looked at how properties of trust have been modeled in them. We have also structured our survey into six focused sections. Not surprisingly the review process was not very straightforward as the reviewed documents did not come readily broken down into the six desired sections! Much deep digging into the model and assumption unraveling had to be carried out. This in itself is a key finding: that current models have

been built with little or no explicit reference to the social properties of trust as uncovered by the huge body of work within the humanities fields. Many of the models reviewed are based on the authors' personal world view of how trust or reputation 'works'. At best, this can only be biased towards the authors' own experiences within his or her environment.

We shall summarise our findings in those six sections below.

4.7.1 Definitions

The question "what is trust" is a difficult one to answer because trust is such a complex and elusive notion. It is thus not surprising to find that the models surveyed, although claiming to handle various aspects of trust, all but two of them failed to define what trust is. Trust is mainly represented as a subjective belief in another entity, except in IST where trust is an objective statistical evaluation. PGP made use of the term 'trusted', but did not explicitly define what it means to be 'trusted' in its model. Four models relates trust to expectations about a trustee's future actions. Lastly, the scope of trust in their models were not given explicitly, except for in X.509, PGP and BBK.

4.7.2 Typology

There are three types of trust; System, Dispositional and Inter-personal trusts. All the models surveyed supported Interpersonal Trust. This is because all the models deal with trust in some other entity, i.e. directly another entity for something. Dispositional Trust is explicitly used in Marsh, but only implicitly in PGP and Reagle. The rest did not have support for Dispositional Trust. Lastly, only one model (Gong) supported System Trust, albeit implicitly.

4.7.3 Properties

Here, we focused on various properties of a trust relationship, namely 1) whether trust is subjective or objective; 2) whether there are more than just two levels of trust supported; 3) whether trust is assumed always to be transitive; and 4) whether agents are allowed to constraint their trust in another to specific contexts only. As shown in Table 4.4, there is an almost unique subset of supported properties for each model. In general, most models treat trust as something subjective, with varying degrees, transitive and constrainable. Further detail are in §4.3

4.7.4 Non-positive Trust

What we are interested here is in if and how the models consider 1) relationships involving distrust; 2) entities with an unknown level of trustworthiness (ignorance); and 3) what happens when a trusted agent reneges on the trust (mistrust). All but six of the models allow their users to specify, in one form or another, whom they distrust. Only IST, Jøsang, Marsh and PGP support ignorance. To be able to handle mistrust, the model must be designed to receive feedback from interactions and act on them. Only three of those surveyed are such models (IST, Marsh and RJ), and all three handle mistrust.

4.7.5 Representation

To be able to implement the trust models, a concrete representation of trust is required. Those that conceptualise trust as binary construct (existence or non-existence of a trust relationship) use different representations. In general, they are all statements made by agents of the form "A trusts B about X". For models that handle various degrees of trust, six represent trust as continuous values between 0 or -1 and 1, and eight used discrete levels for trust. The discrete levels differ from one model to the next, with some using a bounded range and others allowing the value to extend to infinity. Table 4.6 and Table 4.7 contain details about the binary and continuous representations respectively.

4.7.6 Dynamics

Since all models work with trust relationships at some point, all of them have detailed designs of how a trust relationship is built. However, only four of them, BBK, IST, Marsh and RJ, make use of feedback information to revise the level of trust in the trustee. The models can be divided into those that handle trust formation automatically in the model, those that allow manual assertions only and those that allow both. The majority of the models allow both (see Table 4.11 for details).

Chapter 5

Model Outline and Key Concepts

We need a framework not just for protecting privacy but for generally measuring trust – tools and subsystems within the Net that can help individual users make practical decisions about the trustworthiness of specific websites.

Charles Jennings and Lori Fena, founders of TRUSTe

In the next five chapters (chapters 5-10) we describe the design of Ntropi and formally specify its properties. It should be stressed that the design of Ntropi is in no way definitive. The choices we have made in designing our trust model are tentative solutions, aimed at showing how social trust properties *can* be instantiated in computer models, i.e. just one of the many possible ways of modeling social trust. However, we will discuss the reasoning behind the choices made and justify the design choices, their merits, and their weaknesses within the text.

Essin said that "regardless of how simple or complex the decision making process, trust is always comes reduced to a yes/no answer" [Ess97]. In our view, this process involves two interacting components: 1) the decision making policy, which is application specific, and 2) the reputation evaluator, which aggregates experiences and opinions across applications. Ntropi is aimed at the latter, i.e. it is a framework for evaluating agents' reputation for trustworthiness.

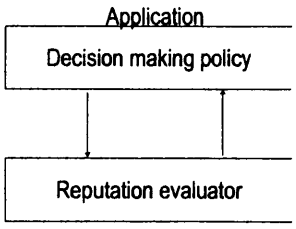


Figure 5.1: Decision making involves two major components: the policy and the reputation evaluator.

Smets and Kennes developed the two-level Transferrable Belief Model (TBM) [SK94, SK97, Sme98]. The first level, called the *credal* level, is functionally similar to the reputation component in Ntropi, in that it is where "beliefs are entertained (and updated)" [SK94]. The second level, called the *pignistic* level, is where decisions are made, based on beliefs at the credal level. This is similar in function to the policy component. However, the similarity ends there as their pignistic level transforms beliefs into probabilities, which isn't the case with our model policy component. However,

```
EXTERNAL TRUST MANAGEMENT POLICY
IF Phase == Unfamiliar
THEN grantService(serviceLevel=limited,pricingLevel=prepay)
WHEN
    FOR Context=cServicePayment
        ExperienceTrust ⩾ +II
        OR
        Reputation ⩾ +I, AND
        ReputationReliability ⩾ +II
    AND
    FOR Context=cCodeReliability
        ExperienceTrust ⩾ 0
        OR
        Reputation ⩾ 0, AND
        ReputationReliability ⩾ +II
    ENDFOR
ENDWHEN
```

Figure 5.2: Example generic application policy that makes use of trust values.

this separation of concerns in this work parallels current thinking in work on belief systems.

The goal of this framework is to provide an autonomous agent with *one* approach to answering the question "How trustworthy has agent X been in situation Y?" for a given time and situation. Although the policy component of the decision process is not within the scope of this work, we will show, through examples made with an example generic policy language, how Ntropi may interface with decision-making policies.

This chapter provides the necessary foundations for the next few chapters where detailed algorithms will be presented. It covers the approach we took in designing the model, the basic concepts involved in our design and the general outline of the model.

5.1 Approach

Current research into trust and reputation modeling is largely focused within specific application domains, such as peer-to-peer file sharing applications or agent-based commerce. This approach is important because the situational dependencies of trust parameters requires analysis of those parameters in order to understand their role in the computation of trust and reputation in those applications.

However, there are important and interesting questions that arise when one looks at how reputational information is exchanged between applications and how the different parameters of trust in each domain make use of the exchanged information. Answering these questions requires work on a more generalised trust model, one where general cross-application properties of trust can be analysed and modeled. We believe that work on the general model of trust is equally important for the reasons outlined above and complements the more focused efforts within each application domain. As such, out work is motivated towards the goal of understanding and building a general, cross-application model of trust.

It is tempting to say that by the end of this thesis we would have described a completely general

model for trust in all possible distributed systems interactions. Alas, such a model must take into account all possible knowledge and all of its possible consequences in all possible users - a job a little heavy for one student and his supervisor! Therefore the work in this dissertation can be considered a subset (of a truly general model) that we think may be useful for interacting agents in distributed systems.

Furthermore, it must be stressed that our goal is not to design a definitive solution, but to suggest, by way of a thought experiment, how desirable properties of a social trust model can be mechanised for use in distributed computer systems.

In our endeavour to design a truly objective model, it will, at best, be based on our own knowledge, disposition and prejudices. Thus this work has been carried out with the position that it will, hopefully, inspire new ideas, be improved upon and evolve into something completely new and much more useful or even indispensable.

In designing this model, we start with the question "Can agent X be trusted?". In other words, we put ourselves in the shoes of an agent about to carry out a certain action which involves interacting with other agents. By working with a generic decision policy as an example to drive the design, we identify how such a policy, or application using it, may make use of trust values. An example generic policy is shown in Figure 5.2

In a real application this would be part of the policy component as shown in Figure 5.1. Thus it is external to Ntropi. The example external policies such as the one in Figure 5.2 tells us the required inputs and outputs for Ntropi, the reputation component.

What we are concerned with here is whether these other agents are *trustworthy* enough for us to go ahead with carrying out the action, or sufficiently fallible so that different measures need be taken. What the different measures are, will depend on the situation involved, and are subjective, hence excluded from this model. An alternative view of this model is as a decision support tool that measures the trustworthiness of parties involved in a particular situation.

The primary output we require from Ntropi will be a value representing the trustworthiness level of a given agent and context. In addition, Ntropi also allows applications to query the phase of an agent's trust relationship, as it may be used to select the appropriate application policy.

As discussed in the social science survey chapters earlier, prior experience and reputation are key pieces of information in the formation and dynamics of trust. To mirror this, in Ntropi, we measure an agent's trustworthiness by looking at two types of information: our **direct experiences** with that agent, and its **reputation** among other agents. In §2.7 we discussed how more experience with an agent allows us to assess his trustworthiness better. For this, some form of learning algorithm or feedback loop will be necessary. We will also need to keep records of experiences so that we can review experiences with the trustee in order to update our relationship with him.

We then look at the semantics and roles of the different types of trust and how they influence one another. Specifically, we determine how the dynamics of each type of trust is affected by any other trust type. Once a general model of trust is developed, we can then provide the specific metrics we would like to use and the algorithms required to process them.

To ensure that the model we design can be engineered and used, we will, where appropriate, augment the descriptive design notes with formal specifications of the model in Z notation. The Z language was chosen because of its widespread use and its adoption by the International Standards Organisation (ISO) as an international standard for software specification [2202]. Being a formal language also means that formal proofs for specifications in Z are straightforward. Furthermore, the author's prior experience with the language has also impacted its preference in this work. For readers unfamiliar with Z, a recommended reference (which is also used as the Z reference for this work) is "An Introduction to Formal Specification and Z" by Potter et al [PST96].

5.2 Basic Concepts

Some of the basic concepts described in this section have been encountered in previous chapters. We will include them in this section again because we would like to define these concepts now in more concrete terms and in the context of our model. Thus, from here onwards, the meanings of the major terms and concepts will be as described below.

5.2.1 Trust

Trust is the central concept in this work.

If a truster, Alice, *trusts* a trustee Bob, then this signifies a passive construct that is synonymous with the phrase "Alice believes Bob is trustworthy". It indicates an opinion of Alice such that, from Bob's general reputation and Alice's experiences with Bob (if any), Alice believes that Bob is trustworthy. Furthermore, this concept is not an absolute one. Trust is a form of belief, as we discussed in Chapter 2. We shall accept that there are degrees of belief. Thus, trustworthiness contains a *degree* element, i.e. Alice trusts Bob with a certain degree d, or Alice believes Bob is trustworthy to degree d. This opinion is subjective.

We will use the verbs **trusting** and **trusted** to refer to actual actions. When we say that Alice trusted Bob, it means that Alice carried out an action that relied on Bob behaving trustworthily. We make no assumptions about Alice's belief at that point of action, merely an observation of that action. Both *trusting* and *trusted* will rarely be used in this dissertation but we include and discuss them here for completeness.

Based on the background research in Chapter 2, our position is that value judgements about trustworthiness cannot be made by a general purpose algorithm. The basis of trust differs from one person to the next and from one culture to the next. This model works with data that represents different levels of trustworthiness, but how that level of trustworthiness is arrived at from a specific observation or experience of an interaction is carried out outside this model. This can be done manually or by algorithms that are tailored for specific application domains, for which specific assertions of what constitute trustworthy or untrustworthy behaviour, and more importantly, their subjective degrees of trustworthiness, can be more effectively made and justified.

For each experience with a trustee, a value representing the quality of the experience in terms of trustworthy behaviour can be assigned by one of those external evaluators and fed into the model as a single *experience level*. These experiences are collated and stored and later summarised using

simple statistical formulae. Additionally, opinions from other agents which are again represented as a particular trustworthiness levels are also collected and stored. The summary of experiences and opinions for a particular agent is that agent's trustworthiness level, as viewed by the truster making that evaluation.

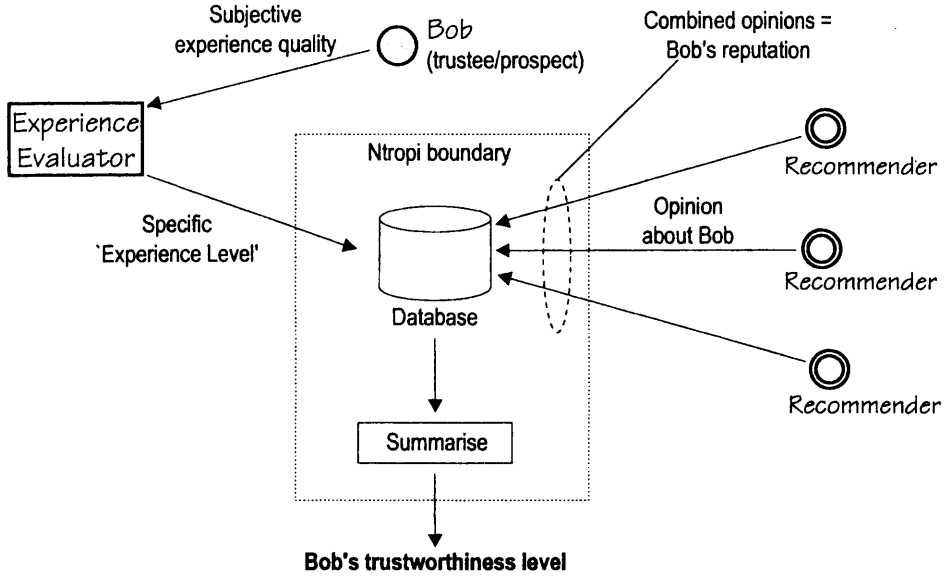


Figure 5.3: Trustworthiness is summary of experience and reputation

5.2.2 Agent

All entities in this model must be able to make decisions, i.e. they must be active entities. An active entity will be referred to as an **agent** to differentiate it from 'passive' entities. Agents in this model must be active because they must be able to decide on actions to be taken and have opinions about other agents' trustworthiness levels. Agents can be human or artificial (software). A keyboard LED, although able to react to 'ON/OFF' instructions, are not agents. They are passive or reactionary entities that merely react to commands or stimuli. We can talk about the LED's reliability, but not its trustworthiness.

We will assume that a non-human (artificial) agent ultimately belongs to an owner who is human (or a group of humans). Thus the artificial agent behaves in accordance with instructions given to it by the human owner. The ownership can be transitive or direct. Ownership by a human is an important concept as it impacts the concept of 'policy', as explained in §5.2.13.

As before, we will use the term *truster* to refer to the agent that trusts, or is considering trusting, another agent. We refer to the agent whose trustworthiness is being evaluated as the *prospect* and once we have decided to trust her she will then be the *trustee*. The terms *distruster* and *distrustee* may also be used in a similar manner to truster and trustee respectively, albeit only in cases of distrust.

5.2.3 Trust Relationship

In our previous surveys, especially in Chapter 4, the term 'trust relationship' is often encountered. The exact semantics of this term, however, differs from one model to another. In Ntropi a **trust relationship** exists between two agents when one agent has an opinion about the other agent's trustworthiness. Thus, unlike previous models, a trust relationship in Ntropi serves more as an encapsulating concept that embodies the opinion about context, phase and level of trust that one agent has in another, rather than an intentional, concrete construct.

A trust relationship has three essential properties:

- A trust relationship can only exist between two agents. However, each agent can be a single entity (e.g. a person) or aggregate entity (e.g. an organisation). While we acknowledge that there are additional group dynamics to consider, in particular, on how a group of agents arrives at a consensus regarding another agent's trustworthiness, in this model we will treat both types of entities as single agents.
- A trust relationship is uni-directional. If Alice trusts Bob and Bob trusts Alice, then we will treat each trust relationship separately.
- A trust relationship is (or rather its properties are) dynamic, so may change over time.

We do not distinguish between 'direct trust' and 'recommender trust' relationships as BBK does. Instead, we separate them within the *context* of the relationship, as will be described in §5.2.11.

5.2.4 Opinion

An **opinion** indicates an agent's belief about another's trustworthiness. Opinions are formed from third party recommendations and from experiences with other agents (see below). Opinions are subjective.

5.2.5 Recommendation

An opinion can also be communicated from one agent to another. For example, Alice may tell Cathy what she thinks of Bob's trustworthiness. Here, Alice has communicated her opinion about Bob's trustworthiness to Cathy. Since our model is designed to work in distributed computer systems, a specific construct or data structure is necessary for handling this communicated opinion. A **recommendation** refers to a construct which contains an agent's opinion.

An agent wishing to communicate his opinion will put the opinion into a recommendation and send it to another agent according to a specific **recommendation protocol**. In this model, an agent requesting an opinion/recommendation from another agent will be called the **requester**, and the agent giving the recommendation the **recommender**.

5.2.6 Reputation

The sum of collected opinions about an agent will be referred to as the agent's **reputation**. A reputation is only visible to the agent that has collected those opinions and not to anybody else, unless

explicitly communicated in a recommendation. For example, Alice receives recommendations (containing opinions about Eric's trustworthiness) from Bob, Cathy and David. Their opinions are now in Alice's 'database' of collected opinions about Eric. These opinions together form Eric's reputation, the subjective content or value of which is only in Alice's eyes. As soon as Alice communicates the content of this reputation, it automatically becomes (to the agent receiving this content) Alice's opinion.

Since reputational information is the result of aggregating opinions from recommenders, the resulting reputation is also subjective because it depends on how much the recommenders are trusted and how the collected opinions are processed into reputational information. Therefore, Alice may have one view of Eric's reputation that differs from Bob's view of Eric's reputation. In DAI parlance, we do not assume that an agent's reputation is objective *common knowledge* [FHMV96].

5.2.7 Trust level

There may be different levels of trustworthiness attributed to a trustee. The semantics, however, depend on the nature of the trust relationship. Specifically, the meaning of a particular trust level depends on whether the relationship has a *direct* context or *recommend* context. The labels for the different trust levels are shown in Table 5.1.

Trust Level	Example descriptive label
+II	Very Trustworthy
+I	Trustworthy
0	Moderate
-I	Untrustworthy
-II	Very Untrustworthy

Table 5.1: Trust level scale.

Our trust level scale is ordinal but the difference between two elements in the scale is mathematically undefined. We cannot assume, for example, that the difference between +II (*Very Trustworthy*) and +I (*Trustworthy*) is equal to the difference between -I (*Untrustworthy*) and -II (*Very Untrustworthy*).

These labels have very loosely associated semantics. Their main purpose is to serve as placeholders for agents to fine tune their semantics with experience and time. Without some kind of global reference scale, it will be difficult to exchange recommendations about trust levels – this 'placeholder' approach serves to facilitate this exchange of opinions. This is discussed further in subsequent chapters when we go into the details of our algorithms. Suffice it to say for now that the trust level of a trust relationship is only meaningful to the truster; it is locally defined.

There is one important distinction between the semantics of the trust levels for a *direct* context and *recommend* context. Details of these two context types are given in §5.2.11 below. Here we will just mention that the trust levels for a direct context (a truster's direct trust in the experiences of another agent) are subjective values given after evaluating the experiences had with the agent, while the trust levels for a recommend context reflect the level of a recommender's consistency in giving his recommendations. We will revisit this distinction when we discuss how trust levels are evaluated.

An additional level of trust we will use is *blind trust* (or *blind distrust*). When an agent chooses to blindly trust or distrust another agent, then no further experience or recommendation can change that

trust level. Since blind trust is manually asserted within a user's policy and does not impact other parts of the model, we do not include it within the 5-level scale.

5.2.8 Semantic Distance

Because each agent's opinion is subjective, it is only prudent that we do not take recommendations at face value. Cathy's meaning of 'trustworthy' may not be equivalent to Alice's understanding of 'trustworthy'. Our individuality makes this difference in value judgement a fact of everyday life. In this model we refer to this discrepancy as the **semantic distance** between two opinions. Semantic distance is important in our model because we are constantly evaluating recommendations and so will need to 'translate' those recommendations into our own understanding of what the recommender means, in terms of our own semantics of the different trust levels.

Formally, the semantic distance between a recommendation of trust level r and its meaning, trust level t, as perceived by the requester is $ord(t) - ord(r)$.

Semantic distances are unique to each recommender, to each context and to each level of trust recommended. So when Cathy makes a recommendation of +I (*Trustworthy*) and Alice takes that as meaning 0 (*Moderate*), or semantic distance = 1, it does not follow that when Cathy makes a recommendation of +II (*Very Trustworthy*) it translates automatically to Alice's +I (*Trustworthy*), also semantic distance = 1, as this time the recommendation may well equate to Alice's +II (*Very Trustworthy*), for example.

Semantic distances are not global values but are local to each agent. They are learnt with each experience of relying on each recommendation and then reconciling the result of the resulting experience had with the recommended trustee with the recommended trust level about that trustee.

5.2.9 Experience

An **experience** is the result of interacting with an agent. When Alice sends her car to Bob the Mechanic for repair work, the quality of that repair work reflects Alice's experience with Bob. Thus, in an experience, the *quality* of that experience is important. That quality is subjective – Alice may find that the job was of excellent quality while another person may think it was poor.

The example above shows the first type of experience, that is the *direct experience*. The other type of experience in this model is called a *recommendation experience*. Imagine that Alice sent her car to Bob after being recommended to Bob by Cathy. In this case, Alice relied on Cathy's recommendation. If Bob did a good job (in Alice's opinion) then she may go back to Bob the next time. The outcome of Bob's repair job affects Alice's opinion about Cathy's future recommendations. In other words, Alice's direct experience with Bob impacts Alice's opinion about Cathy's recommendation – this is the recommendation experience, i.e. the experience of relying on that recommendation.

A recommendation experience is *per recommender*. In cases where more than one opinion (a reputation) affected the truster's decision, the quality of those experiences will still be compared to each individual recommendation to arrive at the recommendation experience for each recommender.

All experiences in this model have an associated experience level whose domain is the trust levels

in Table 5.1. An experience level for a particular interaction is an indication of how trustworthy the trustee behaved in that interaction. This value is subjective and is only meaningful to the truster who had and evaluated that experience. However, the actual evaluation of the experience and assignment of an experience level to that experience is done externally to Ntropi. The reason for not building this evaluation into our model is experiences are highly subjective and it is unreasonable expect to be able to design a general algorithm that can evaluate arbitrary experiences. The only requirement for the external experience evaluation module is that it returns an experience level to our model – how it evaluates the experience is irrelevant.

5.2.10 Database

We will refer to an agent's collected opinions (reputation information) and experiences as the agent's **database**.

5.2.11 Context

Context qualifies a trust opinion, describing what the truster's belief in another's trustworthiness is really about. For example, if Alice trusts Bob with respect to providing stock market advice then "Providing stock market advice" is the context of that trust. All opinions about trust, or trusting actions, must have a context – previous discussions have shown us that without context, trust expressions are, at best, ambiguous.

We discussed the difference between the semantics of direct and recommended trust values in §5.2.7 and the nature of experiences of each type. Consequently, there are two types of contexts: **direct** and **recommend** contexts. The example in the previous paragraph is an instance of direct context, i.e. trust that is based on our direct experiences with the agent. When the trustee is a recommender, the trust relationship exists within a recommend context. For instance, Alice's trust in Cathy's recommendation in §5.2.9 exists within the recommend context. These types of context deal with the different types of relationships in the same way as BBK's direct and recommender trust relationships.

The distinction between these two context types can be defined more concretely in terms of their implementations in our model. For relationships within a direct context, trust is based upon the *typical* experience with the trustee, as observed by the truster himself. In this model, this means taking a statistical measure of central tendency of those experiences. On the other hand, relationships within the recommend context are based upon two factors:

1. The closeness of a recommender's recommended trust value to our own judgement about the direct trustee's trustworthiness, i.e. the semantic distance between our own experience and the recommended trust value, and

2. The recommender's reliability because we would like to know which recommenders are more reliable in the quality of their recommendations, within a given period of time.

The first measure, semantic distance, is used for initial encounters with new recommenders, in order to determine the closeness of his opinion to our own. If the minimum level of closeness is met then the relationship can continue. As the relationship progresses, the level of trust during the lifetime of

this relationship is then based on the second measure, reliability, which is taken as the measure of the spread of semantic distances for that recommender. In our model, we use the semi-interquartile range (SIQR) to measure this spread. The SIQR is suitable for qualitative data such as the trust levels in Ntropi.

In our notations we will discriminate these two context types by using d or rn ($n \in \mathbb{N}_1$) prefixes to the context variable c , thus $d.c$ will signify a direct context and $rn.c$ will signify a recommend context.

Assume that Alice had not known David before and was looking for someone to recommend a stock market expert and had first asked a friend, Cathy, to recommend someone. Cathy could not come up with a recommendation so she referred Alice to David instead, which is how Alice came to know David. David finally recommends Bob to Alice as a stock market expert. Cathy is in effect also a recommender for the same recommend context, but Cathy's role is slightly different from David's – while David is a recommender of the expert (i.e. Bob), Cathy is a recommender of a recommender. It is easy to generalise this to have recommenders of recommenders of recommenders, ad infinitum. But, from our discussion on page 98, we learned that two different recommendation chains require two different contexts of trust – a one-size-fits-all-chain-length recommendation trust is not feasible. Therefore, we must recognise each recommender's ability to recommend different chain lengths differently. In this model, we do this by using an indexed prefix; if n is the chain length then $rn.c$ is the context for that recommendation chain. So $n = 1$ is one level of recommendation (e.g. David above, with context $r1.c$), $n = 2$ is two levels of recommendation (e.g. Cathy above, with context $r2.c$), and so on. $r0.c$ is equivalent to $d.c$.

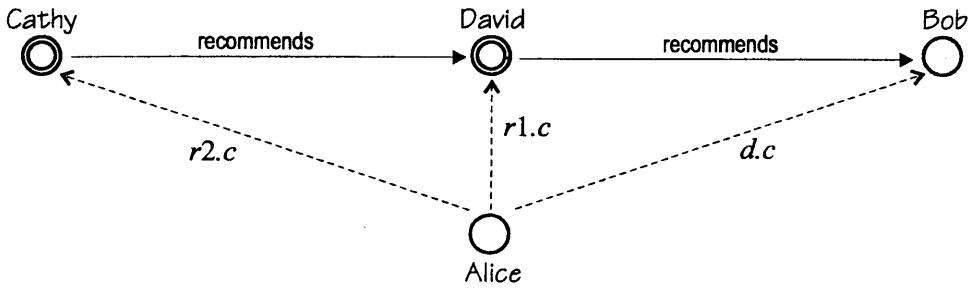


Figure 5.4: Recommend context. Shown is a recommendation chain from Cathy to Bob and Alice's relationship context (dotted arrows) with each of the agent in the chain with respect to the shown chain.

Context Type	Symbol	Description
Direct	$d.c$	Context for general trust in other agents.
Recommend	$rn.c$	Context for recommender trust in other agents in their capacity as recommenders. n is an integer variable that signifies the chain length of the recommendation. $n > 0$.

Table 5.2: Context types.

Whenever we describe one agent's belief in another's trustworthiness, or one agent's trusting actions towards another, an active context is assumed.

It is also possible that two or more contexts are related in such a way that an agent's trustworthiness

in one context says something about his trustworthiness in another. For example, if we trust an agent for references to good academic papers on a particular subject, we may also trust him for references to researchers in the same field. We base this latter trust on the assumption that knowing about the papers entails also knowing about authors of those papers, who are researchers themselves. Context inter-relationships are not modeled in this work due to time constraints but they are a very interesting area highly relevant to trust modeling. Hence we discuss this further in the Future Work chapter of this dissertation (see Chapter 12).

5.2.12 Typology

As we know from the survey in Chapter 2, there are three different types of trust that may be active at any time; *Interpersonal Trust*, *Dispositional Trust* and *System Trust*. In this model, we will incorporate Interpersonal Trust and Dispositional Trust.

System Trust is a recent topic in the social sciences and its mechanisms and dynamics are still poorly understood when compared to the other trust types above. As a consequence there is little information that we can use for designing System trust components in our model in this work. As such, System Trust is beyond the scope of this work.

Interpersonal Trust is concerned with an agent's reputation. Its dynamics is supervised by Dispositional Trust which dictates how trust is granted initially, how fast trust is built and how fast it is destroyed. In this model, the agent's policy regarding various parameters of the model will be the manifestation of its Dispositional Trust (see §5.2.13 below).

These types of trust are summarised in the table below which shows the different trust types in the model and what they relate to.

Trust Type	Trust subject
Interpersonal	Another agent.
Dispositional	Decision affecting the dynamics of Interpersonal trust.

Table 5.3: Trust types in the model.

5.2.13 Policy

As observed in §5.2.12 above, an agent's disposition is contained in its *policy*. An agent's policy governs such things as the various threshold parameters used in the model and the dynamics of Interpersonal Trust. In general, an agent's policy contains values that can only be manually inserted, modified or removed. For software agents, this means that the policy must be directly maintained by its human owner (either directly 'by hand' or through some other interface or automated mechanism). We imagine a software agent's policy to actually be its owner's policy, specified for the agent. This is consistent with our view that all non-human agents are acting on behalf of some person or organisation (i.e. have human owners).

Unlike other models we have encountered before, *an agent's Dispositional Trust and its policy are one and the same thing.*

5.2.14 Phases

In our model, a trust relationship goes through phases. At any point in time, a trust relationship will exist in one of four phases, as shown in Figure 5.5.

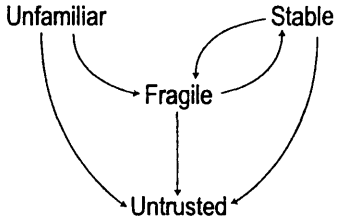


Figure 5.5: Phases of a trust relationship, with arrows indicating possible direction of phase transition.

If an agent, say Alice, does not have an opinion about Bob's trustworthiness, then we say that Alice is unfamiliar with Bob (or Bob is unfamiliar to Alice). Their trust relationship is in the **Unfamiliar** phase. After the first experience, this relationship may either enter the Fragile phase or Untrusted phase, depending on the level of experience and the governing phase transition policy. This mirrors the 'first impressions' strategy we employ in social interactions.

Once Alice begins to have an opinion about Bob's trustworthiness, then Alice's trust relationship with Bob will enter a **Fragile** phase where trust is still tentative and low-risk actions are typically taken. In this phase, it can be very easy for Bob's trustworthiness level to deteriorate towards a very low trust level or even distrust. This is because a trustee's trustworthiness is continuously tested in the Fragile phase until the truster is satisfied that the trustee's trustworthiness has reached a level whereby a more stable trust relationship can be maintained.

The next is the **Stable** phase. Once in this phase, strong reasons will be required to lessen the level of trust in Bob. Put another way, negative experiences have smaller impact in this phase compared to the Fragile phase. Nevertheless, an agent can specify a threshold for the ratio of negative to positive experiences beyond which a Stable relationship can fall back into the Fragile or Untrusted phase.

Currently, there is no way out of the **Untrusted** phase, apart from the user making a manual change. Future work will investigate 'forgiveness' mechanisms whereby an agent can learn to re-trust untrusted agents.

As we discussed in §2.7, relationships at the Stable phase require less monitoring from the truster. It also means that the truster can rely more on his own judgements about the trustee, so there will be less need to enquire about the trustee from other agents (the truster's recommenders). These factors mean that the cost of maintaining a Stable relationships is less than that of a Fragile relationship.

For example, in situations where the request for and transmission of recommendations may be costly, such as when the time required to wait for recommendations to arrive is anticipated to be beyond which a trusting decision must be made, a stable trust relationship with the prospect can give the truster a higher confidence level when trusting the prospect, without additional recommendations from third parties. In other words, in situations where reputational information is not available, identification of stable trust relationships (that are typically based on positive history) reduces un-

certainties and increases confidence when faced with a trust decision.

Phase transitions can change an Unfamiliar phase to a Fragile phase but not vice versa. However Fragile can change to Stable and vice versa.

All phases are context specific. For example, Alice may have a Stable relationship with Bob within the "Car Mechanic" context but a Fragile one with respect to the "Investment Advice" context. The initial amount of trust to grant unfamiliar agents and the quality of experiences relevant to the dynamics of each phase will be determined by the truster's policy, or Dispositional Trust.

In this work, we may sometimes refer to a trust relationship in the Fragile phase as a Fragile Trust Relationship and those in the Stable phase as Stable Trust Relationships.

5.2.15 Names

For reputations to exist, the model must allow agents to be referenced in a consistent way. We need to be able to name agents. It would be impossible to form a reputation about an agent without having a reference to that agent. Names were discussed in greater depth in §3.8.

An agent need not only have one name, and an agent need not have a globally unique identifier. What is important is that within a truster's local namespace, its names or references to other agents are unique. This property of names also indicates that some means of translating names between the local namespaces of agents must be provided. One method of handling this is proposed in the SDSI specification, which is now incorporated in the SPKI standard [E+99]. Names can also be in the form of pseudonyms (or 'nyms'). Freenet [CMH+02] is an example system that uses this approach.

To simplify matters, we will assume that a name in a recommendation is understood to refer to the same agent by both the requester and recommender. In practice, the implementer of this model may include an interface to an external name-translation mechanism such as the one employed in SPKI or other out-of-band mechanisms.

There will be no restrictions on names being transferred anonymously between agents, allowing one agent to acquire the reputation of another. This is common practice in commerce where, for instance, brand names are taken over by another company after a merger or buyout.

5.2.16 Attributes

When a prospect's reputation is unknown and no experience about him is available, the truster may need to resort to stereotyping the prospect, a method we will cover in the next chapter. This makes use of the prospect's various attributes, like the prospect's current domain.

There are two types of attributes in this model, **implied attributes** and **certificate attributes**. An implied attribute is a 'fact' about the prospect that the truster can determine or guess directly. The prospect's current domain and his operating system are examples of implied attributes. Certificate attributes are digitally signed statements (containing assertions about the prospect's attributes) about the prospect. This is useful in making assertions about affiliations, e.g. prospect is a University College London staff member, or qualifications, e.g. prospect is an accountant certified by the Chartered Institute of Management Accountants.

Anybody can issue a certificate – all the issuer has to do is digitally sign a statement. It is up to the truster to determine whether the digital signature is valid (belongs to whom the truster thinks the signer is) and how much the truster trusts the issuer with respect to the statement made.

5.3 General Model Outline

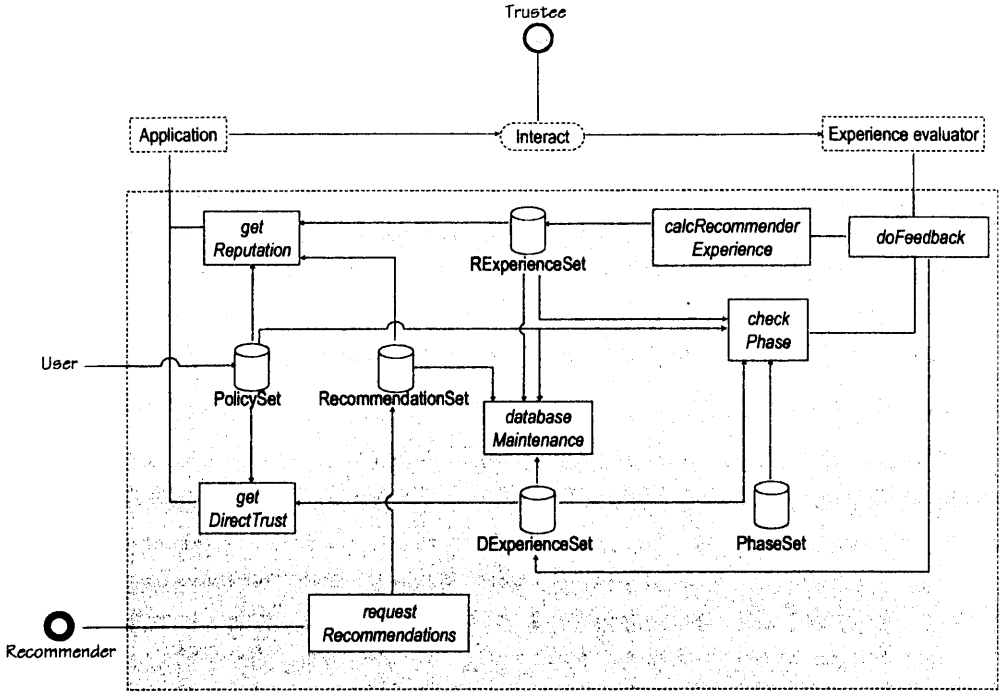


Figure 5.6: General outline of the Ntropi framework. Shaded area indicates the scope of the model.

Figure 5.6 is a high level illustration of Ntropi, showing the various tasks and databases involved. The items outside of the shaded area are outside the scope of Ntropi.

In general, an application using the Ntropi trust model will call either the *getReputation* or *getDirectTrust* functions, or both, depending on the policy defined within the application. The functions' parameters also include the prospect's ID and the context of trust. The trust management policy used in the application may look like the example shown in Figure 5.2.

Both *getReputation* and *getDirectTrust* calculate interpersonal trust (see §2.2). *getReputation* returns a reputation value for the given agent ID and context, as well as a measure of reliability of the returned value, based on the trustworthiness of the recommenders that contributed to that reputation value. The information used to calculate these values are obtained from the recommendations obtained from recommenders, stored in the RecommendationSet database, and past experiences with the contributing recommenders, stored in the RExperienceSet database. Reputation algorithms are discussed in Chapter 8.

getDirectTrust returns a value representing the prospect's trustworthiness based on past experiences with him. This makes use of past direct experiences stored in the DExperienceSet database. Chapter 6 covers direct trust evaluations.

How the algorithms in the two *calc* functions above interpret past experiences and recommendations is governed by various parameters set in the Ntropi's policy database, PolicySet. PolicySet also contains policies regarding phase transitions and maintenance of databases, particularly when to discard old database entries. PolicySet mirrors dispositional trust in social models of trust (see §2.4.4).

requestRecommendations is responsible for sending out requests for recommendations to various recommenders, receiving them and storing them in the RecommendationSet database. This is carried out according to a recommendation protocol. Chapter 9 discusses the recommendation protocol.

After an interaction with an agent (also called the trustee), an external experience evaluation algorithm rates the experience with the trustee. This is then fed back into Ntropi so that the trustee's state, and the state of any recommenders that contributed recommendations about him, can be updated. This is done by calling the *doFeedback* functions. *doFeedback* in turn stores the experience level passed to it into the DExperienceSet database, evaluates the experiences with any recommenders that were involved in calculating the trustee's reputation and finally checks the phase of the trustee's and recommenders' relationships to determine of any of them needs to be changed in the light of the new experience. Trustee phases are stored in PhaseSet.

Finally, a database maintenance function, *databaseMaintenance* is available for housekeeping work, like purging old recommendations and experiences once past its "use-by" threshold, as determined in PolicySet.

5.4 Chapter Summary

This chapter describes the basic concepts involved in this model and the assumptions made when designing it and outlined the general model. The major contributions of this chapter are:

- A *5-level trust scale* was introduced (§5.2.7).
- Trust relationships exist within a specific *context*. Contexts are divided into *direct* or *recommend* contexts to reflect the nature of the trustee in the relationship (§5.2.11).
- Ntropi handles two types of trust: *Interpersonal* and *Dispositional* (§5.2.12).
- Every trust relationship may go through two or more of the four possible *phases*, which are the *Unfamiliar*, *Fragile*, *Stable* and *Untrusted* phases (§5.2.14).

We describe the model's outline and the high-level process of calculating trust and evaluating feedback.

We hope to have laid the necessary foundations for the reader go forward to the next few chapters where we will look at the model's algorithms in greater detail.

Chapter 6

First Encounters

Be slow to fall in friendship; but when thou art in, continue firm and constant.

Socrates

Judging the trustworthiness of a person we have just met or an organisation we are dealing with for the first time is a normal everyday occurrence. During our initial conversation or interaction with this 'stranger' we ask questions and look for cues of trust or distrust. A trust 'cue' would be something that is familiar to us, and which triggers a sense of trust (or distrust) towards that stranger (this was discussed in §2.8.2). These cues are used to judge trustworthiness because we do not have any "history" with this new person upon which we can call.

We learn these cues from experiences in current and past trust relationships with other agents. From these relationships we select cues that we feel may be closely related to the new prospect and situation that we currently find ourselves in, i.e. those that match visible properties we receive from the prospect. This method effectively groups, or "stereotypes", the prospect with previous trustees who share the same properties. The amount of trust we grant the group will be the same used to grant the prospect.

To help us work through algorithms in Ntropi we will use a running example of a peer-to-peer (P2P) application model. In our example P2P model all nodes are peers in the decision making process, that is, each node makes its own decisions about who to trust and interact with, unlike hierarchical models where perhaps the network administrator has control of computers in his workgroup or domain and makes decisions on whom to trust.

6.1 Running example

Figure 6.1: The peer-to-peer processor sharing application model as running example.

The example application we will run over our example P2P network is one that allows idle processor time on a user's computer to be shared with others on the network, somewhat similar to Grid systems [FK99]. Idle processor time can be used to run other users' code, for free or for a fee.

As shown in the diagram above, Bob wants to share his processor so that idle time on his processor time can be used by others. It turns out that Alice has some processing to do and would like Bob to run some of her code to speed up her work. We ignore for now how Alice came to know Bob and his processor sharing service.

The scenario so far already raises some security issues and trust questions that each node will have to consider. Questions that Bob may ask about Alice include:

- How reliable is the code?
- Is Alice malicious?
- Will Alice pay for the service on time?

Answers to his questions will allow Bob to make an informed decision on whether to allow Alice access to his processor sharing service or not. Furthermore, Bob can decide to use this information to make finer grained decisions. For example, if Bob decides that Alice is trustworthy enough to use the service, and depending on *how much* he trusts Alice, he may choose to grant different levels of service and select a suitable pricing policy. The former may include limits on processor cycles and amount of data generated; the latter may include credit-based or pre-paid charging schemes.

In his decision making, Bob will require to determine Alice's trustworthiness. Bob has two options for doing this:

1. Try to determine trust based on his own experiences.
2. Try and find out Alice's reputation from his trusted sources.

In this chapter we will look at the first option and see how Bob can decide on his own. Option two will be discussed in Chapter 8. Before we continue, we will first try and understand how the results of these evaluations will be used in Bob's decision making process.

6.2 Policies and Trust Decision Management

Bob's questions above regarding Alice's trustworthiness tells Bob which contexts are relevant in this example. From Bob's questions previously he can decide that the relevant trust contexts for this particular interaction with Alice are as follows:

- Code reliability, *d.cCodeReliability*.
- Non malicious behaviour, *d.cNonMalicious*.
- Prompt payment for service, *d.cServicePayment*.

```
EXTERNAL TRUST MANAGEMENT POLICY
IF Phase == Unfamiliar
THEN grantService(serviceLevel=limited,pricingLevel=prepay)
WHEN
      FOR Context=cServicePayment
            DirectTrust < +II
      AND
      FOR Context=cCodeReliability
            DirectTrust < +II
      AND
      FOR Context=cNonMalicious
            DirectTrust < +II
      ENDFOR
ENDWHEN
```

Figure 6.2: Example External Policy that uses recommended trust values.

In this instance, Bob has identified three relevant contexts. We will use the identifiers in italics for reference to these contexts. Bob may decide that some of these contexts are more important than others. This 'weighting' of contexts will depend on Bob's policies regarding this interaction. How the actual weighting works and whether the contexts are combined in the decision making process in any way is beyond the scope of this work. This is because such tasks are specific to each area of application and are therefore very subjective. For example, the decision making process for medical diagnosis is quite different from that of credit assessment.

However, there are tools in existence that one can use for such tasks. Examples of these 'trust management' tools include KeyNote [BFI+99], SULTAN [Gra03], PICS [KMRT96] and REFEREE [CFL+97]. These tools allow trust policies to be formed and evaluated. Such a policy may include a minimum level of trustworthiness required before trust is granted. Since the task of Ntropi is essentially to answer the question "How trustworthy is prospect P given the context C", it complements the trust management tools above.

The example in Figure 6.2 is a generic trust-management policy that mirrors the way the tools above work. It is an example of how Bob may define his policy for granting or rejecting access to his processor sharing service. This example states the conditions under which Bob will allow a prospect pre-paid use of his processor for a limited amount of time. Thus, Alice's trustworthiness with respect to each of the three contexts must be evaluated and a minimum level for each must be met before the policy statement succeeds and the service is granted.

A trust-management tool whose duty it is to enforce the policy above would make three calls to Ntropi, each corresponding to one of the *DirectTrust* conditions in the example policy in Figure 6.2. Specifically, each of these calls are made to the function *getDirectTrust*. Ntropi will then return a trustworthiness value for the given context and principal name.

A trust evaluation in Ntropi begins with the function call to *getDirectTrust*(*prospect*, *context*) for a given prospect and context. Within *getDirectTrust*, the method of evaluation depends on the current phase of the relationship with *prospect*. The result of the evaluation is a single value of type *TRUSTVALUE*.

$$TRUSTVALUE ::= -II \mid -I \mid 0 \mid +I \mid +II$$

AGENTID is a basic Z type for a reference to an arbitrary agent. In an actual implementation this could be a string representing a name or a network address, such as an IP address. *CONTEXTID* is another basic type and it represents a reference to a context.

$$[AGENTID, CONTEXTID]$$

The function *getDirectTrust* is defined below.

getDirectTrust

$$forProspect? : AGENTID$$
$$forContext? : CONTEXTID$$
$$result! : TRUSTVALUE$$

$$getPhase(forProspect, forContext) = phUnfamiliar \Rightarrow result! = getUnfamiliar(forProspect?, forContext?)$$
$$getPhase(forProspect, forContext) = phFragile \Rightarrow result! = getFragile(forProspect?, forContext?)$$
$$getPhase(forProspect, forContext) = phStable \Rightarrow result! = getStable(forProspect?, forContext?)$$
$$getPhase(forProspect, forContext) = phUntrusted \Rightarrow result! = -II$$

All *getDirectTrust* does is check the current phase of the trust relationship with the prospect *forProspect* and context *forContext* and returns the result of the appropriate function. The *getPhase* function returns the phase, of type *PHASE*, and the functions *getUnfamiliar*, *getFragile* and *getStable* carries out evaluations for relationships in the Unfamiliar, Fragile and Stable and Untrusted phases. For the untrusted phase, *getDirectTrust* returns the most negative trust value, in this case -II. This is arbitrary, as we expect the phase Untrusted to be sufficient for selecting the appropriate action without knowing how untrustworthy the agent is.

$$PHASE ::= phUntrusted \mid phUnfamiliar \mid phFragile \mid phStable$$

We have separated the trustworthiness calculations for each of the phases into different functions because, from earlier discussions, we have identified that the dynamics of trust are different within each of these phases.

The set *PhaseSet* contains all of Bob's trust relationships and it is made up of elements of type *PhaseRec*.

PhaseRec

$$agent : AGENTID$$
$$context : CONTEXTID$$
$$currentPhase : PHASE$$
$$since : DATETIME$$

$$PhaseSet == \mathbb{P}\, PhaseRec$$

DATETIME is a basic type representing the date and time:

$$[DATETIME]$$

The *getPhase* function is defined as follows:

getPhase

$$\Xi PhaseSet$$
$$prospect? : AGENTID$$
$$currContext? : CONTEXTID$$
$$result! : PHASE$$

$$\forall er : PhaseRec \mid (er.agent = prospect?) \land (er.context = currContext?)$$
$$\bullet ((er \in PhaseSet) \Rightarrow (result! = er.currentPhase))$$
$$\lor ((er \notin PhaseSet) \Rightarrow (result! = phUnfamiliar))$$

6.3 Evaluating a Prospect Based on Own Experiences

Being unfamiliar with Alice means that Bob does not have any experience with Alice. Because of this lack of direct information about Alice, Bob will now have to resort to other information to *indirectly* estimate Alice's trustworthiness. This could be recommendations from his trusted recommenders about Alice, or internal information from Bob's own experiences with other agents, but it will be information about Bob's experiences within the same context as Alice's. We will delay discussion of recommendations until Chapter 8. For now, we will look at how Bob can make use of his past experiences with other agents to formulate an opinion about Alice's trustworthiness. Later we will look at how this can be combined with evaluations on Alice's reputation, giving Bob a more informed opinion about Alice's trustworthiness.

What Bob will need to do is collect those past experiences within the same context as that in which he encounters Alice and summarise that set of experiences. There are two classes of generalised information, called *Classifiers*, which Bob can call upon for when evaluating Alice in this manner: **Context Experience** and **Stereotype**. Table 6.1 details these two generalisation classifiers.

Classifier	Description
Context Experience	General trustworthiness of all other trustees we have experienced in the current context and use this as a basis for a 'typical behaviour' for trustees in this context.
Stereotype	Groups past trustees based on a common attribute with the prospect and summarises the general trustworthiness of those trustees. We then form an opinion about those trustees as a group and include the prospect in that group, effectively transforming our opinion about the group into an opinion about the prospect.

Table 6.1: Generalised information classifiers for first encounters.

These classifiers correspond to the categorisation mechanisms used in social trust, as discussed in our social science survey, page 51. The context experience classifier is an implementation of the reputation categorisation mechanism and the stereotyping classifier is an implementation of the stereotyping mechanism.

The choice of which classifier to go for will depend on Bob's Ntropi Policy assertions. We represent a truster's set of policies with *PolicySet*. Assume that we have a basic generic type *POLICY* which represents the content of a policy assertion.

$$[POLICY]$$

We then have a policy record, which represents the type for each element in *PolicySet*.

PolicyRec

$$policyName : STRING$$
$$policyContext : CONTEXTID$$
$$policyContent : POLICY$$

$$PolicySet == \mathbb{P}\, PolicyRec$$

To add or modify a policy, the function *addPolicy* is used, passing it the policy name and content and parameters:

addPolicy

$$\Delta PolicySet$$
$$newPolicyRec? : PolicyRec$$

$$PolicySet' = PolicySet \oplus \{newPolicyRec?\}$$

In our example, Bob asserts that for prospects in the *d.cServicePayment* context, Ntropi is to evaluate using the Context-Experience classifier, and if that fails to return a value, to use the Stereotype classifier. This is carried out thus:

$$addPolicy(npClassifier, d.cServicePayment, (clContextExperience, clStereotype))$$

For querying policies, we will use a generic function *getPolicy*. It is generic in terms of the result returned (i.e. of type POLICY) because, as we shall see, policies exist in various forms, thus it will be limiting at this point to define concrete data structures for each type. We will assume that actual implementations will carry out the appropriate type conversion.

getPolicy

$\Xi PolicySet$
$forPolicyName? : STRING$
$forContext? : CONTEXTID$
$result! : POLICY$

$\forall p : PolicyRec \mid (p.policyName = forPolicyName?) \land (p.policyContext = forContext?)$
$\quad \land (p \in PolicySet) \Rightarrow result! = p.policyContent$

The job of the *getUnfamiliar* function is then to query this policy and call the right classifier function according to it.

getUnfamiliar

$forProspect? : AGENTID$
$forContext? : CONTEXTID$
$val : TRUSTVALUE$
$result! : TRUSTVALUE$
$csPolicy : POLICY$

$csPolicy = getPolicy(npClassifier, forContext)$

$\text{head}csPolicy = clContextExperience \Rightarrow$
$\quad val = getContextExperience(forProspect?, forContext?) \land$
$\quad val \neq undefined \Rightarrow result! = val \land$
$\quad val = undefined \Rightarrow result! = getStereotype(forProspect?, forContext?)$
$\text{head}csPolicy = clStereotype \Rightarrow$
$\quad val = getStereotype(forProspect?, forContext?) \land$
$\quad val \neq undefined \Rightarrow result! = val \land$
$\quad val = undefined \Rightarrow result! = getContextExperience(forProspect?, forContext?)$

6.4 Context Experience

To use Context Experience, all Bob has to do is look in his database of past experiences for those within the context in which he is interested (say, in this case, *d.cServicePayment*) and summarise this information to get a single general trustworthiness value. The result should be Bob's own opinion about the general trustworthiness of trustees he had encountered within this context.

To 'summarise' this information essentially means that Bob has to look at the data and try to formulate a hypothesis based on that data. Now there are many statistical algorithms that exist on how this should be carried out. However, the method we must choose for Bob to use will be based on our initial investigations into how humans make these decisions in the real world. The model of generalised trust developed by Hardin, as discussed in §2.7.1, is a well referenced model. In his model, new trust relationships are based on the trustee's trusting disposition. The trustee's disposition is an internal aversion based on accumulated experiences within a particular context. As Hardin said, "experience moulds the psychology of trust" [Har93].

In Ntropi, as in Hardin's model, an agent's disposition when encountering new agents is an indication of whether he is generally a trusting or a distrusting agent. In other words, it depends on whether he is risk-averse, risk-selective or somewhere in between when it comes to granting trust. In Ntropi, these dispositions are strategies that an agent can choose to employ in various trust evaluation situations. The following are the different strategies available to Bob in our example when it comes to generalising past experiences. The preferred disposition is specified under the Ntropi Policy called General Trust Disposition (GTD). Assume that in this example, Bob asserts the following GTD's.

addPolicy(*npGenTrustDisp*,cServicePayment, *gtdMedian*)
addPolicy(*npGenTrustDisp*,cCodeReliability, *gtdMedian*)
addPolicy(*npGenTrustDisp*,cNonMalicious, *gtdRiskAverse*)

Trustee	Experience level
a_1	0
a_2	+I
a_3	+I
a_4	-I
a_5	+I
a_6	0
a_7	+II

Table 6.2: Example: Bob's previous experiences in the *cServicePayment* context.

Table 6.2 is an example showing Bob's past experiences for the *d.cServicePayment* context. With the Risk-Averse strategy, the worst experience had will be selected, i.e. -I, and then returned to the decision making process, as we have seen in the previous section. Selecting the worst experience ensures that the least trust possible is granted (so that the truster is being risk-averse). The Risk-Selective strategy simply selects the best experience had, in this case, +II. This will ensure the most trust possible is given, reflecting the higher risk in this strategy.

The previous two strategies deal with the extremes of a truster's disposition. However, most of us would probably opt to make a more 'realistic' analysis of our experiences and decide to go with the

majority of experiences we had, or a moving average over time. For Bob, one such measure would be the statistical median of the sample of his past experiences in the active context, i.e. the set of experiences shown in Table 6.2, giving the value 1: $median(\{0, +I, +I, -I, +I, 0, +II\}) = +I$. At times, the statistical mode may be used as alternative to median as it gives a good measurement of the most frequent experience encountered. However, mode poses problems when the data set is multi-modal. Thus the reasons for using the median measurement are because: 1) median is not affected by multi-modal data and 2) it gives the same result as mode when the data set is unimodal. We now present these algorithms formally.

Past direct experiences are stored in the *DExperienceSet* set:

$$DExperienceSet == \mathbb{P}\, DExperienceRec$$

where ExperienceRec is the data structure that contains a record of one experience, defined as follows:

DExperienceRec

$timestamp : DATETIME$
$context : CONTEXTID$
$agent : AGENTID$
$experienceLevel : TRUSTVALUE$

We can now define the *getContextExperience* function which returns the result of the Context Experience generalisation classifier for a given context:

getContextExperience

$\Xi DExperienceSet$
$forContext : CONTEXTID$
$result! : TRUSTVALUE$
$ContextExperienceSet : \mathbb{P}\, TRUSTVALUE$

$$ContextExperienceSet == \{ \forall er : DExperienceRec \mid (er.context = forContext) \land (er \in DExperienceSet) \bullet er.experienceLevel \}$$

$$(getPolicy(npGenTrustDisp, forContext) = gtdRiskAverse) \Rightarrow (result! = min(ContextExperienceSet))$$
$$(getPolicy(npGenTrustDisp, forContext) = gtdRiskSelective) \Rightarrow (result! = max(ContextExperienceSet))$$
$$(getPolicy(npGenTrustDisp, forContext) = gtdMedian) \Rightarrow (result! = median(ContextExperienceSet))$$

6.5 Stereotypes

The Stereotypes classifier is the other generalisation alternative for Bob. 'Stereotyping' would usually carry a negative connotation but it is an everyday strategy that we employ when faced with uncertainties and to avoid the more costly (usually time-wise) alternative of searching for information about a prospect before entering into an interaction. We discussed the social science account of stereotyping in §2.7.1.

To use the Stereotypes classifier, Bob will need to group together past trustees with whom he has or had relationships with within the *d.cCodeReliability* context and summarise this information.

Grouping is based on an attribute common to all trustees — we will call this the *grouping-attribute*. We will only consider single attributes for now. Extending the model to filter on multiple attributes will be in future work. All grouping attributes are observable objective facts about agents. Facts include things like e-mail addresses or date of birth. This is to differentiate it from subjective attributes like honesty and kindness.

Bob first compiles whatever attributes of Alice that he can find and stores them in the *prospectAttribs* set. For each of Alice's attributes, Bob will do the following:

1. Find all of his past trustees whom Bob had experiences with in the active context, that have the same attribute value.
2. For each of those trustees, calculate their trustworthiness level based on experiences had and reputations (calculating trustworthiness will be presented in the next few chapters).
3. From the list of calculated trustworthiness levels for each trustee above, apply the General Trust Disposition to determine the trustworthiness level to associate with this attribute.

The result is that Bob will now hold a list of possible stereotypical trustworthiness levels. If there is more than one value in this list then his General Trust Disposition is then applied to the values in that list to arrive at the final single trustworthiness value for Alice.

Recall from §5.2.16 that implied attributes are pieces of information that the trustee determines directly, like domain address (from the prospect's email address) and operating system type (from the prospect's request message), and certificate attributes are assertions by a third party about the prospect. Certificate attributes can be obtained by requesting a certificate from a third party or by requesting a certificate from Alice herself. Because the credibility of the credentials depends on the trustworthiness of the agent that digitally signed (hence issued) the certificate, certificates need not necessarily be obtained only from Bob's trusted sources. All certificates must be evaluated for their credibility. Implied attributes are valid and credible by default as it was Bob who implied these attributes[1].

Assume that Bob managed to imply from Alice's email address that her request is from the `ucl.ac.uk` domain. Further, Bob also obtained a certificate signed by UCL saying that Alice is the Head of Department in the Biology Department.

[1] We acknowledge that confidence values can be used to determine the confidence the truster has in the implied attributed, but we will not cover it's use in this work.

Attrib name	Attrib Value ($\{V\}_S$: message V signed by agent S)
Domain	ucl.ac.uk
Employee	$\{$Name=Alice, Pos=Head-of-Dept, Dept=Biology$\}_{UCL}$

From Bob's own attributes database, he found the following attributes of his previous trustees that are the same as Alice's, with the trustworthiness values arrived at after calculations according to the algorithm above.

Agent	Attrib name	Attrib Value	Trustworthiness
a_1	Domain	*.ac.uk	0
a_2	Domain	*.ac.uk	+I
a_5	Domain	*.ac.uk	+I

Table 6.3: Example: Bob's database of trustees' attributes.

The results for the three different Trust Dispositions would then be 0 if the General Trust Disposition is risk averse, $+I$ for risk selective and $+I$ for median.

The following is the algorithm in formal notation. We will take *Attribute* as a basic type which represents an arbitrary attribute of an agent.

$[ATTRIBUTE]$

Any attribute about known agents are stored in the *AgentAttributeSet* whose elements are records of the type *AttribRec*:

AttributeRec

$agent : AGENTID$

$attrib : ATTRIBUTE$

$AgentAttributeSet == \mathbb{P}\, AttributeRec$

The *getStereotype* function for querying stereotype experience evaluation is thus given below.

The first line of the *getStereotypes* schema is a condition that ensures the prospect's attributes have been stored in *AgentAttributeSet* before *getStereotype* is called. This is so that the prospect's attributes can be found by *getStereotypes* using the *getAttributes* function below. The *getAttributes* function returns a set of attributes (each element of type *ATTRIBUTE*) for a given prospect and stores them in *ProspectAttributeSet*.

Next, the set of agents that share the same attributes as the prospect is created (the *SimilarAgentSet* set). For each of the agents in *SimilarAgentSet*, their direct trust levels are obtained and stored in PossibleResultSet. Note that this will not be a recursive call as the agents in SimilarAgentSet are known agents, and therefore will not be evaluated as Unfamiliar agents, hence this *getStereotype* function will not be called again.

The final value is selected according to the General Trust Disposiion as defined in the Ntropi Policy.

getStereotype

$\Xi AgentAttributeSet$
$prospect? : AGENTID$
$currentContext? : CONTEXTID$
$result! : TRUSTVALUE$
$ProspectAttributeSet : \mathbb{P}\, ATTRIBUTE$
$SimilarAgentSet : \mathbb{P}\, AGENTID$
$PossibleResultSet : \mathbb{P}\, TRUSTVALUE$

$\exists\, ar : AttributeRec \bullet (ar.agent = prospect?) \land (ar \in AgentAttributeSet)$

$ProspectAttributeSet = getAttributes(prospect?)$

$SimilarAgentSet = \{ \forall\, prospectAttrib, agentAttrib : AttributeRec \mid$
$\quad (prospectAttrib \in ProspectAttributeSet) \land$
$\quad (agentAttrib \in AgentAttributeSet) \land$
$\quad (agentAttrib.agent \neq prospect?) \land$
$\quad (prospectAttrib = agentAttrib.attrib) \bullet agentAttrib.agent \}$

$PossibleResultSet = \{ a : AGENTID \mid a \in SimilarAgentSet$
$\quad \bullet\, getDirectTrust(a, currentContext?) \}$

$(getPolicy(npGenTrustDisp, currentContext?) = gtdRiskAverse)$
$\quad \Rightarrow (result! = min(PossibleResultSet))$
$(getPolicy(npGenTrustDisp, currentContext?) = gtdRiskSelective)$
$\quad \Rightarrow (result! = max(PossibleResultSet))$
$(getPolicy(npGenTrustDisp, currentContext?) = gtdMedian)$
$\quad \Rightarrow (result! = median(PossibleResultSet))$

getAttributes

$\Xi AgentAttributeSet$
$forAgent? : AGENTID$
$result! : \mathbb{P}\, ATTRIBUTE$

$result! = \{ \forall\, ar : AttributeRec \mid (ar.agent = forAgent?) \land (ar \in AgentAttributeSet)$
$\quad \bullet\, ar.attrib \}$

The function *addAttribute* simply adds a new attribute record for an agent into the *AgentAttributesSet* set:

addAttribute

$\Delta AgentAttributeSet$

$newAttributeRec? : AttributeRec$

$AgentAttributeSet' = AgentAttributeSet \cup \{newAttributeRec\}$

If a matching attribute value cannot be found in past trustees, then Bob will have to decide manually. In this case, Bob can make a manual assertion in his Ntropi Policy regarding various attributes and decide what trust values to grant. For example, if the above processing fails to generate a trust value and assuming that Bob already trusts the UCL registry, he may assert this in the Ntropi Policy:

$$addPolicy(npBlindTrust, (attrib, \text{'ucl.ac.uk'}, context, +II))$$

This is Ntropi's Blind Trust Policy where, in this example, prospects with requests originating from the 'ucl.ac.uk' domain are automatically granted trust level $+II$ by Bob.

Notice that we are currently assuming a generic type for attributes, *ATTRIBUTE*. In real applications we envisage a range of attribute types from different applications and situations, e.g. email addresses and digitally signed certificates. Thus we assume that the equality operator '=' involving variables of *ATTRIBUTE* will have to be overridden according to the actual data type of the attribute.

Future work will allow user to specify which attributes to use for a given context.

6.6 Putting It Together

We now look at how the algorithms presented previously tie together, using our running example with Bob. Assume that the following is Bob's complete policy statement for service provision, defined externally to Ntropi using a form of Trust Management tool, such as KeyNote. We will assume that the policy enforcement engine tries to satisfy each policy in sequence until it finds one that succeeds or until all fails, somewhat akin to the execution of Prolog[2] or any declarative programming language. Bob's policy below has been modified and enriched with finer grained criteria involving phases.

The first policy statement states that if a request originates from one of Bob's email addresses then he should grant the request. The granted service level should not have any limits on the execution time and there shall be no charge.

The second policy statement states that for relationships in the Unfamiliar or Fragile phases, no service is to be granted if any of the prospect's trustworthiness levels for *d.cCodeReliability*, *d.cNonMalicious* and *d.cServicePayment* are less than 0, +I and 0 respectively.

The third and fourth statements are defined in a similar manner. A *serviceLevel* = *limited* assertion

[2]Prolog is the implementation language used in the SULTAN trust management system [Gra03].

EXTERNAL TRUST MANAGEMENT POLICY

```
IF (prospect == bob@bobshomedomain.com)
OR (prospect == bob@bobsworkdomain.com)
THEN grantService(serviceLevel=unlimited, pricingLevel=free)
ENDIF

IF Phase == Unfamiliar OR Fragile THEN
grantService(serviceLevel=no-service,pricingLevel=none)
WHEN
      FOR Context=cServicePayment, DirectTrust < 0
      AND FOR Context=cCodeReliability, DirectTrust < 0
      AND FOR Context=cNonMalicious, DirectTrust < +I
ENDWHEN
grantService(serviceLevel=limited,pricingLevel=prepay)
WHEN
      FOR Context=cServicePayment, DirectTrust < +II
      AND FOR Context=cCodeReliability, DirectTrust < +II
      AND FOR Context=cNonMalicious, DirectTrust < +II
ENDWHEN
ENDIF

IF Phase == Stable THEN
grantService(serviceLevel=limited,pricingLevel=prepay)
WHEN
      FOR Context=cServicePayment, DirectTrust < +I
      AND FOR Context=cCodeReliability, DirectTrust < +I
      AND FOR Context=cNonMalicious, DirectTrust < +II
ENDWHEN
grantService(serviceLevel=limited,pricingLevel=credit)
WHEN
      FOR Context=cServicePayment, DirectTrust = +II
      AND FOR Context=cCodeReliability, DirectTrust = +II
      AND FOR Context=cNonMalicious, DirectTrust = +II
ENDWHEN
ENDIF
```

Figure 6.3: Example External Policy that uses recommended trust values.

indicates that the service is limited by some threshold, by number of processor cycles for example. A *pricingLevel* = *prepay* indicates that the prospect is required to pay for the service before he can use it, and a *pricingLevel* = *credit* allows processes to be run and then the owner billed later, which is useful for those trustees whom Bob already knows and trusts.

Bob receives a request from Alice to use his idle processor time for running some Java code. The request was received from the email address `alice@bio.ucl.ac.uk`.

Bob has told Ntropi to evaluate using the Context-Experience strategy first and if that fails, to use the Stereotype strategy by using *addPolicy* to assert these conditions (page 145).

Assuming that for the *d.cCodeReliability* context, Bob was not able to find any previous trustees' experiences for this context. This means that Ntropi will have to evaluate this using the Stereotype classifier. Using the result shown in §6.5 above and applying the Median disposition as in Bob's Ntropi policy, we get the final value of +I for Alice's trustworthiness value for this context.

For the *d.cServicePayment* context Bob managed to find some previous experiences. The calculation in §6.4 gives the result of +I, so this will be the assumed trustworthiness level for Alice in this context.

Finally, we will assume that the resulting trustworthiness value for Alice for the third context, *d.cNonMalicious* equates to 0 after evaluation using one of the two Ntropi classifiers.

The final result of Ntropi's evaluation for the three contexts is shown in the table below:

Context	Trustworthiness value
d.cCodeReliability	+I
d.cNonMalicious	0
d.cServicePayment	+I

This matches the second service provision policy under the Unfamiliar or Fragile phases conditions which states that a limited pre-paid service can be granted to Alice. Thus this is the course of action taken by Bob once Ntropi has returned the three trustworthiness values required back to the Trust Management tool which is evaluating the policies.

6.7 Experience Evaluation and Feedback

Assume that Alice has been granted the limited pre-pay service and at this point her Java program had just finished running on Bob's processor. This is the moment when Bob is able to generate opinions about Alice's trustworthiness based on the experience he just had with her. For our example we will assume that Bob has experienced the following about Alice:

- That she paid on time and her money cleared.
- That her program did not try to do anything malicious.
- That her code seemed to have run without any problems.

So all in all Alice, and her Java program, have behaved well and seem worthy of trust in the future. Bob can now record this experience in his direct experiences set *DExperience*:

EXTERNAL TRUST MANAGEMENT POLICY
IF paymentReceivedDate > 30 days THEN
FOR Context = cServicePayment, experienceLevel = -II

Figure 6.4: Example experience evaluation policy.

$$DExperience' = DExperience \oplus \{(\text{10-08-04 11:06:35,d.cServicePayment}, Alice, 1),$$
$$(\text{10-08-04 11:06:35,d.cCodeReliability}, Alice, 1),$$
$$(\text{10-08-04 11:06:35,d.cNonMalicious}, Alice, 1)\}$$

This can be carried out with repeated calls to the *addExperience* function, defined as follows:

addExperience

$\Delta DExperienceSet$
$newExperienceRec? : DExperienceRec$

$DExperienceSet' = DExperienceSet \cup \{newExperienceRec?\}$

The experience values are evaluated externally to Ntropi. As discussed in §5.3, this is because of the subjective and application specific nature of experiences. For this example however, we can imagine that an external experience evaluation policy may look like the one shown in Figure 6.4.

6.8 Chapter Summary

In this chapter we discussed Ntropi's model for evaluating unfamiliar agents, i.e. first encounters. Algorithms were outlined formally in the Z specification language.

We started with a running example of where an agent, Bob, provided a processor time sharing service to online customers. This example will be used and expanded upon when required to illustrate various algorithms throughout this dissertation.

Working from example trust management policies from the running example, we identified the function calls and parameters that Ntropi must provide. The entry point is the function *getDirectTrust*, which calls the appropriate subroutines based upon the phase of the given agent for the given context. We then outlined the function *getUnfamiliar*, the main function for unfamiliar agents.

For unfamiliar agents, Ntropi evaluates trust values based on generalised past experiences with other known agents. Based on the social "reputation categorisation" mechanism, Ntropi provides two methods for doing this, the choice of which depends on which information *classifier* is selected in the Ntropi Policy.

The *context experience* classifier aggregates past experiences for the given context and returns a single trust value. The returned value is a function of the set of past experiences and the truster's general trust disposition for the given context. The algorithm is detailed in the function

getContextExperience.

The *stereotype* classifier attempts to base the prospect's trustworthiness on other agents the truster has encountered who share the same attributes as the prospect. Opinion is thus based upon the prospect as a member of the shared-attribute group. The algorithm for stereotyping is given in the function *getStereotype*.

To illustrate these algorithms we worked through a full example at the end of the chapter, looking at how the various functions tie together.

Chapter 7

Evolving Relationships

The ultimate test of a relationship is to disagree, but hold hands.

Alexander Penney

In the last chapter we discussed the model for interacting with strangers with examples from the initial encounter to post-interaction evaluation of the experience. Following the evaluation and recording of the experience, the truster can also evaluate the phase of the relationship and decide whether it should evolve into another phase. In this chapter, we will look at how a relationship changes from one phase to another, or its *phase transition*, and discuss experience trust evaluation for relationships in the Fragile and Stable phases.

Recall that a trust relationship may be in one of four phases at any one time: Unfamiliar, Fragile, Stable or Untrusted (see §5.2.14). The rules that define phase changes are asserted by the truster in the Ntropi Policy using *thresholds* for each experience value.

Gambetta [Gam88a] and Luhmann [Luh79] both suggested that threshold mechanisms play a part in trust – they provide simples methods for managing the complexity of trust decisions. The notion of trust as threshold was discussed in §2.4.3.

In Ntropi, *thresholds* define triggers for when a phase transition should take place for a trust relationship. A single threshold value for a particular experience level indicates the minimum number of experiences for that level to be reached before phase transition takes place. For example, if the threshold for experience level -I is 3 then when the truster had 3 experiences of level -I with the trustee, a phase transition evaluation will be executed. This is specified per context. Thresholds can be specified for one or more experience levels in each context. Phase transition is carried out when any one of the threshold is met, i.e. the thresholds defined for each experience level for a context are disjunctively combined, or OR-ed. For instance, if the thresholds for context c are m for level -I and n for level -II, then a phase transition is triggered when the truster has had either m experiences with the trustee of level -I OR n experiences of level -II.

With the threshold parameter, Bob, in our running example, can assert the following in his Ntropi Policy:

For any Fragile relationship in the cServicePayment context, change the relationship to

untrusted if the trustee has produced either 1 experience at level -II or 3 experiences at level -I.

In the example assertion above, the thresholds are 1 for experience level -II, and 3 for experience level -I. We can represent this as a set of exactly five tuples where a tuple (l, t) represent a threshold of t for the experience level l. Thus the example assertion above would be represented as {(-II,1),(-I,3),(0,0),(+I,0),(+II,0)}. A zero threshold means 'ignore this experience level'. The policy concerns transition from Fragile to untrusted. Lastly, the context is *cServicePayment.*

We represent this policy data structure with the *PTPolicyRec* schema. The predicate portion of this schema is a uniqueness pre-condition which says that only one threshold per experience level is allowed.

PTPolicyRec

$$context : CONTEXTID$$
$$fromPhase, toPhase : PHASE$$
$$ThresholdSet : \mathbb{P}\, TRUSTVALUE \mapsto \mathbb{N}$$

$$\forall tv_1, tv_2 : TRUSTVALUE;\ m, n : \mathbb{N} \bullet$$
$$((tv_1, m) \in ThresholdSet) \land ((tv_2, n) \in ThresholdSet) \Rightarrow tv_1 \neq tv_2$$

The *PTPolicySet* set contains all of the truster's policy on phase transitions:

$$PTPolicySet == \mathbb{P}\, PTPolicyRec$$

The function *addPTPolicy* is to facilitate asserting new phase transition policies:

addPTPolicy

$$\Delta PTPolicySet$$
$$newPTPolictRec? : PTPolicyRec$$

$$PTPolicySet' = PTPolicySet \oplus \{newThresholdSet?\}$$

Thus, to add a phase transition policy, the following call can be made:

$$addPTPolicy(\mu\, PTPolicyRec \mid context = d.cServicePayment \land fromPhase = phUnfamiliar$$
$$\land\ toPhase = phUntrusted \land ThresholdSet = \{(-II, 1), (-I, 0), (0, 0), (+I, 0), (+II, 0)\})$$

Assume that Bob creates the phase transition policies as shown in Table 7.1. As an example to show what the table means, the third row of the table indicates that if the trustee in the Fragile phase (column 1) generates one experience at level -II (column 3) or three at level -I (column 4) then the relationship phase will change to Untrusted (column 8).

From Phase	To Phase	Thresholds				
		-II	-I	0	+I	+II
Unfamiliar	Untrusted	1	0	0	0	0
Unfamiliar	Fragile	0	1	1	1	1
Fragile	Untrusted	1	3	0	0	0
Fragile	Stable	0	0	20	10	5
Stable	Fragile	5	10	0	0	0
Stable	Untrusted	7	15	0	0	0

Table 7.1: Example Phase Transition Policy for context *cServicePayment*.

We saw in the previous chapter that the last experience Bob had with Alice in the *cServicePayment* context was of level +I. Therefore, according to his phase transition policy in Table 7.1, this changes his relationship phase with Alice for this context to Fragile. To do this, we just need to ensure the entry corresponding to this relationship in the *PhaseSet* set contains this new phase, or if an entry has not yet been created, to add one. The *setPhase* function facilitates this:

setPhase

$$\Delta PhaseSet$$
$$newPhaseRec? : PhaseRec$$
$$PhaseSet' = PhaseSet \oplus \{newPhaseRec?\}$$

The preceding process is presented more formally in the *checkDPhase* function schema below.

checkDPhase

$$forTrustee? : AGENTID$$
$$forContext? : CONTEXTID$$
$$trusteePhase : PHASE$$
$$EffectivePolicySet : \mathbb{P}\, PTPolicyRec$$
$$trusteePhase = getPhase(forTrustee?, forContext?)$$
$$EffectivePolicySet = getPTPolicy(forContext?, trusteePhase)$$
$$\exists\, ep : PTPolicyRec \bullet$$
$$ep \in EffectivePolicySet$$
$$(\wedge\, \exists\, threshold : TRUSTVALUE \mapsto \mathbb{N} \mid threshold \in ep.ThresholdSet \bullet$$
$$\#\{\, \forall x : TRUSTVALUE \mapsto TRUSTVALUE \mid$$
$$x \in ((\mathrm{id}\, getExperiences(forTrustee?, forContext?, trusteePhase.since))$$
$$\lhd first(threshold)) \bullet first(x)\,\} = second(threshold))$$
$$\Rightarrow setPhase(forTrustee?, forContext?, ep.toPhase)$$

The function *getPTPolicy* in the second line of *checkDPhase*'s predicate above returns the set of phase transition policies relevant to a given context and prior phase.

getPTPolicy

$\Xi PTPolicySet$
$forContext? : CONTEXTID$
$forPhase? : PHASE$
$result! : \mathbb{P}\, PTPolicyRec$

$result! = \{ ptrec : PTPolicyRec \mid (ptrec \in PTPolicySet)$
$\quad \wedge (ptrec.context = forContext?) \wedge (ptrec.fromPhase = forPhase?) \}$

The function *getExperiences* called in *checkDPhase*'s predicate returns a set of experience levels for a given agent and context which are not older than the given freshness value.

getExperiences

$\Xi DExperienceSet$
$forTrustee? : AGENTID$
$forContext? : CONTEXTID$
$since? : DATETIME$
$result! : \mathbb{P}\, TRUSTVALUE$

$result! = \{er : ExperienceRec \mid er \in DExperienceSet \mid$
$\quad (er.agent = forAgent?) \wedge (er.context = forContext?) \wedge$
$\quad (er.timestamp \geq since) \bullet er.experienceLevel \}$

Imagine that Alice's previous experience with Bob was a positive one so much so that she again approaches Bob for purchase of more processor time. Bob looks up his External Policy as on page 151 and, after a check of the prospect's phase, determines that the second policy block (under 'Unfamiliar or Fragile') matches Alice's phase, i.e. Fragile.

The conditions within the policy block will then be evaluated, prompting calls to the *getDirectTrust* function (see page 141). Assume the following is for the context *d.cServicePayment*. The *getPhase* function call for *Alice* and *d.cServicePayment* will return *phFragile* as recorded in Bob's *PhaseSet* (which contains phases for all his trust relationships). This implies a call to the *getFragile* function, which returns the trust value for *Alice* within context *d.cServicePayment*. We look at this function below.

7.1 Fragile Trust Phase

In the full evaluation of an agent's trust value, previous experiences and reputation information from other agents are evaluated and combined. In this section we will look at the former, i.e. *direct trust*, and defer discussion of the latter until we cover reputation in Chapter 8.

To obtain the direct trust value of the prospect is just a matter of summarising the experiences

the truster had with that prospect. The summarising algorithms are familiar at this point as they are similar to the three context experience strategies discussed in the previous chapter (see §6.4). However, the algorithms are applied to the set of experiences from the prospect in this case. Thus, the algorithm is:

1. Collect all previous experiences with prospect, within the given context.
2. Look up the General Trust Disposition (GTD) for this context to determine summarising function.
3. Summarise, using min, max or median, for the dispositions Risk Averse, Risk Selective or Median, respectively.

Assume that Bob has had the following previous experiences with Alice in the *d.cServicePayment* context: {+I,0,+I}. According to Bob's GTD for this context, *gtdMedian* was specified (see the *addpolicy* examples on 145). Thus the result, i.e. Alice's experience trust value for context *d.cServicePayment*, is $median(\{+I,0,+I\}) = +I$. This is shown formally below:

getFragile

$$\exists DExperienceSet$$
$$forProspect? : AGENTID$$
$$forContext? : CONTEXTID$$
$$ExpWithProspect : \mathbb{P}\, TRUSTVALUE$$

$$ExpWithProspect == \{ \forall er : ExperienceRec \mid (er.agent = forProspect?) \land (er.context = forContext?) \land (er \in DExperienceSet) \bullet er.experienceLevel \}$$

$$(getPolicy(npGenTrustDisp, forContext) = gtdRiskAverse) \Rightarrow (result! = min(ExpWithProspect))$$
$$(getPolicy(npGenTrustDisp, forContext) = gtdRiskSelective) \Rightarrow (result! = max(ExpWithProspect))$$
$$(getPolicy(npGenTrustDisp, forContext) = gtdMedian) \Rightarrow (result! = median(ExpWithProspect))$$

7.2 Stable Trust Phase

Recall that in a stable relationship there is a stronger 'bond' between the parties in the relationship and it takes a higher degree of bad experiences to reduce trust in the relationship (see §2.7.2).

To reflect the differing impact of negative experiences in stable relationships, we will allow different experience levels to have different impacts on our summarisation process. We do this by first assigning weights to the range of experience levels. Each weight defines the degree of impact a particular experience level exerts in relation to the other experience levels. In other words the weightings reflect the ratio of the impacts of the various experience levels.

For example, Bob may have the weightings assigned to various experience levels for the *cServicePayment* context as shown in Table 7.2 below. Experience levels -II and -I have the same impact, level 0 has three times more impact than -II and -I, level +I has four times more impact than -II and -I and level +II has six times more impact than -II and -I, or two times more impact than level 0.

Experience level	-II	-I	0	+I	+II
Impact weighting	1	1	3	4	6

Table 7.2: Example experience impact weighting for stable phase relationships.

During evaluation of a prospect's trustworthiness in the Stable phase, the weighting is applied by multiplying the number of previous experiences by the weight assigned to the level of that experience. This is done for experiences in the active context only. In our example, assume that Bob's relationship with Alice is now in the Stable phase and his set of past experience levels with Alice for the *d.cServicePayment* context is:

$$\{-II, -I, -I, -I, 0, 0, 0, +I, +I, +I, +I, +I, +II, +II, +II, +II, +II\}$$

Applying Bob's impact weightings given in Table 7.2 produces the following weighted experience set:

$$\begin{aligned}
&\{-II,\\
&-I, -I, -I,\\
&0, 0, 0, 0, 0, 0, 0, 0, 0,\\
&+I, +I, +I, +I, +I, +I, +I, +I, +I, +I, +I, +I, +I, +I, +I, +I, +I, \mathbf{+I}, +I\\
&+II, +II, +II, +II, +II, +II, +II, +II, +II, +II, +II, +II, +II, +II, +II, +II,\\
&+II, +II, +II, +II, +II, +II, +II, +II, +II, +II, +II, +II, +II, +II\}
\end{aligned}$$

The Stable Phase experience trust value is then taken from this weighted set by simply selecting the Median value, which is $+I$ (shown in bold in the example set above). We do not use the Risk Averse or Risk Selective strategies in this case because the risk-bias disposition is reflected in the assignment of the weightings.

Experience level	Assigned weight	Frequency of experiences	Weighted frequency
-II	a	v	$a \times v$
-I	b	w	$b \times w$
0	c	x	$c \times x$
+I	d	y	$d \times y$
+II	e	z	$e \times z$

Table 7.3: Application of the weights for stable phase evaluation.

Weightings are stored in the *StableWeightingSet* which contains tuples (*context,weightings-set*) where *weightings-set* is the set of weightings relevant for *context*. Weightings can be assigned and queried by the functions *setWeighting* and *getWeighting* respectively.

$$WeightingSet : \mathbb{P}\, TRUSTVALUE \mapsto \mathbb{N}$$

$$\forall ev_1, ev_2 : TRUSTVALUE;\ n : \mathbb{N} \bullet ((ev_1, n) \in WeightingSet) \land ((ev_2, n) \in WeightingSet) \Rightarrow ev_1 \neq ev_2$$

$$StableWeightingSet == CONTEXTID \nrightarrow WeightingSet$$

setWeighting

$$\Delta StableWeightingSet$$
$$forContext? : CONTEXTID$$
$$newWeightings? : WeightingSet$$
$$StableWeightingSet' = StableWeightingSet \oplus \{(forContext?, newWeightings?)\}$$

getWeighting

$$\Xi StableWeightingSet$$
$$forContext? : CONTEXTID$$
$$result! : WeightingSet$$
$$\exists w : WeightingSet \mid (forContext?, w) \in StableWeightingSet \Rightarrow (result! = w)$$

Experience trust value evaluation, as discussed in the last few paragraphs, is obtained by calling the *getStable* function, shown below:

getStable

$$\Xi DExperienceSet$$
$$forProspect? : AGENTID$$
$$forContext? : CONTEXTID$$
$$result! : TRUSTVALUE$$
$$Weights == getWeighting(forContext?)$$
$$BagOfExp == items\,(seq\,\{\forall er : ExperienceRec \mid (er.agent = forProspect?) \land (er.context = forContext?) \land (er \in DExperienceSet) \bullet er.experienceLevel\})$$
$$\forall w : TRUSTVALUE \mapsto \mathbb{N} \in Weights \bullet first(w) \sqsubseteq BagOfExp \Rightarrow first(w) \sqsubseteq BagOfExp' \land BagOfExp' \natural first(w) = BagOfExp \natural first(w) * second(w) \land BagOfExp' = items\,(seq\,R) \land result! = median(R)$$

The weightings used here, as exemplified in Table 7.2, are arbitrary in nature. At best, this method allows a first approximation of a user's subjective view of the different impacts of experience levels in a given application domain. However, given time, these weightings can be changed as the sensitivity of each experience level is learnt. More importantly however, the individual weighting variables, acting as placeholders for the weighting values themselves, allow these subjective weight distributions to be tuned for each application. This is what we are trying to model here.

7.3 Chapter Summary

In this chapter we described the algorithms for handling phase transition and evaluating direct trust for prospects in the Fragile and Stable phases.

Phase transitions is implemented using a threshold mechanism. Thresholds are specified for the number of various experience levels and this governs the changeover from one phase to another. The threshold scheme mirrors the social trust strategy of applying thresholds to trust decisions, as discussed in §2.4.3. Phase transitions are carried out in the *checkDPhase* function.

For Fragile and Stable phase relationships, trust values are based on taking the median of past experiences within a given context. The difference between the Fragile and Stable phase algorithms is that in the latter, weights can be assigned to the different experience levels. This is to allow the social phenomenon where negative experiences in Stable relationships have less impact than during the Fragile phase.

Chapter 8

Recommendations and Reputation

> *People would not trust someone completely based only on his reputation. But reputation would be important as a screening device in cases when one wonders whether to socialize with someone one does not know.*
>
> *Toshio Yamagishi [Yam98]*

In the last two chapters we looked at how direct trust is evaluated in Ntropi. Direct trust is only half the picture of a prospect's trustworthiness, the other consisting of opinions from the truster's recommenders. In this chapter we will look at how we evaluate third party opinions, or *recommendations*, about a prospective trustee. A number of recommendations combined together form the *reputation* about the prospect. This reputation, combined with experience trust, provides the complete foundation of a prospect's trust value within Ntropi (see Figure 5.3). Later in the chapter we will discuss how experiences with recommenders can be used to inform future recommendations from them.

As in the previous chapter, we outline the decision-making model that the truster will be using in the shape of an external trust management policy (Figure 8.1). This is to aid us in designing a usable recommendation and reputation model. Reputational information is used by the truster in the absence of experience trust. This information will be used to determine how to grant a particular level of service or whether to participate in a transaction.

In Ntropi, the truster is able to use two types of third party information in his policies: the reputation (i.e. aggregated recommended trust levels) for the prospect, and a measure of reliability for the reputation. The latter can be used to select risk levels based on the uncertainty of the recommended trust level. Figure 8.1 is an example of a policy that incorporates recommendations.

Recall that there are two categories of contexts in Ntropi: *direct* and *recommend* contexts (see §5.2.11). If Bob's relationship with an agent Richard is in the recommend category of the *cServicePayment* context then this relates to Bob's opinion about Richard's trustworthiness in giving recommendations (opinions about other agent's trustworthiness) in the *cServicePayment* context. We will differentiate these contexts by using a depth index prefix of the form *rn* for each context name, where $n \in \mathbb{N}$. In a recommendation from Richard to Bob about Craig for context *cServicePayment* (Figure 8.2), Bob's trust relationship with Richard, with respect to this recommendation, is in the context *r1.cServicePayment*. Bob's context of relationship with Richard

```
EXTERNAL TRUST MANAGEMENT POLICY
IF Phase == Unfamiliar
THEN grantService(serviceLevel=limited,pricingLevel=prepay)
WHEN
    FOR Context=cServicePayment
        ExperienceTrust ⩾ +II
        OR
        Reputation ⩾ +I, AND
        ReputationReliability ⩾ +II
    AND
    FOR Context=cCodeReliability
        ExperienceTrust ⩾ 0
        OR
        Reputation ⩾ 0, AND
        ReputationReliability ⩾ +II
    ENDFOR
ENDWHEN
```

Figure 8.1: Example External Policy that uses recommended trust values.

(*r*1.*cServicePayment*) is referred to as Richard's *incontext*, and the context of Richard's recommendation (*r*0.*cServicePayment*) is Richard's *outcontext*. The *r*0 context category is actually equivalent to the direct context: *r*0.*cServicePayment* = *d.cServicePayment*. Notice that Bob's outcontext is equal to Richard's incontext. Thus, in a recommendation from a recommender to a truster, on in a relationship between a truster and trustee:

$$outcontext(truster) == incontext(recommender/trustee)$$

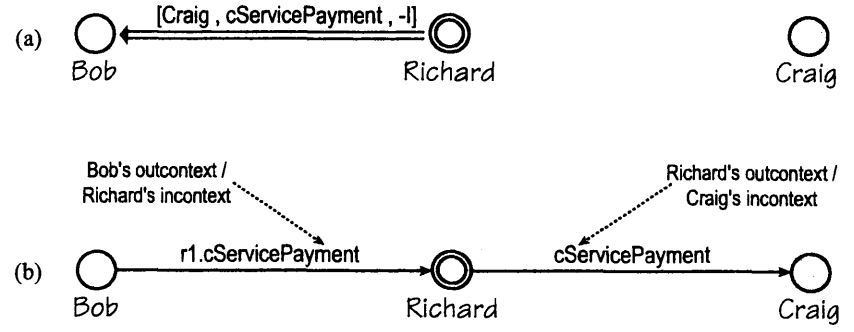


Figure 8.2: (a) Bob receives recommendation from Richard about Craig. (b)This results in the trust relationships and in/outcontexts as shown.

The *n* index in the *rn* prefix is the context's *context depth index*. When $n = 0$ then we say that the relationship or recommendation has context depth index of 0 (this is equal to the direct context). If a context has a depth index of 1, that means the trustee is a recommender of a direct prospect, like the context of the relationship between Bob and Richard above. Note also that if a trustee's incontext has a depth index of 0 then his outcontext is undefined; depth indexes are never less than 0. Higher depth indexes are possible in recommendation chains, discussed later in §8.5. A recommender's incontext always has a depth index of its outcontext plus 1.

$$depthindex(incontext) == depthindex(outcontext) + 1, \qquad depthindex(outcontext) \geqslant 0$$

We define the Z schema *Context* to encapsulate the context name and depth, for use in recommendations.

Context

$contextID : CONTEXTID$

$depth : \mathbb{N}$

Extending our running example, let's assume that Bob is approached by an unfamiliar agent Craig who, like Alice, wants to purchase some processor time. As before, Bob's External Trust Management Policy for his processor sharing service, as shown in Figure 8.1 requires that the prospect's trust levels for the various contexts be evaluated. In Craig's case, it is the Unfamiliar phase conditions that are of concern. We now look at how reputation and reliability measures are evaluated. We start by looking at algorithms for a single direct recommendation before evaluation chains. This is followed by discussions on methods for combining several recommendations.

8.1 Evaluating a Recommendation

Assume that Bob receives a recommendation from Richard about Craig's trustworthiness with respect to the *d.cServicePayment* context, and that Richard is one of Bob's trusted recommenders for this context. We defer discussions on how Bob obtains such a recommendation until Chapter 9 and take it for now that the recommendation from Richard has reached Bob.

A recommendation contains the following information:

- **Timestamp**: When the recommendation was made.
- **Recommender**: The agent who made this recommendation.
- **Prospect**: The agent whom the recommendation is about.
- **Context**: The context name and depth the prospect is being recommended for.
- **Trust level**: The trustworthiness of the subject, in the recommender's opinion.

Field	Content
Timestamp	10-01-04 14:36:09
Recommender	Richard
Prospect	Craig
Context ID	cServicePayment
Context depth	0
Trust level	-I

Table 8.1: Example recommendation from Richard about Craig, context *cServicePayment*

Imagine that Richard's recommendation is as in Table 8.1. Given this recommendation, can Bob rely on it for making decisions? Does he trust Richard sufficiently to make recommendations that he can rely on? Is he confident that when Richard recommends a certain trustworthiness level n he means the same n according to Bob's value standard, or is it higher or lower than Bob's own interpretation of n? These are the questions that Bob will be asking about the recommendation before using it to make decisions. The first task is to know what the recommender means when he says 'trust level n'. This relies on the concept of the *semantic distance*.

8.2 Semantic Distance

In §5.2.8 we introduced the concept of *semantic distance*. The semantic distance between a recommendation for prospect p of trust level r and the actual experience t had with p by the truster who used that recommendation is $ord(t) - ord(r)$. This distance exists because of the difference in the recommended trust value and the outcome of the experience resulting from reliance on that recommendation. This outcome may be due to errors in the recommender's judgement or difference in standards when evaluating the outcome of experiences because agents act autonomously. As we discussed in Chapter 2, trust is subjective. Therefore, opinions about an agent's trustworthiness may differ from one observer to another. When the result of this observation is transmitted and shared, discrepancy in opinion about the same event may result. This discrepancy is what we are trying to model with the semantic distance concept.

There will be uncertainty at the early stages of the relationship with a recommender because it is uncertain whether the semantic differences in the first, second and other early recommendations will remain stable. However, the more recommendations we get from a recommender, and the more we rely on them, the more we will know about the consistency of the semantic distances between our own experiences and the recommended values. This follows the principle that trust is not static but is dynamic and is learnt (§2.7).

Thus semantic distance forms the basis of our trust in a recommender in two ways: firstly, by allowing us to adjust, or *translate*, the recommended value into our own 'value system', and secondly to determine the consistency of his recommendations as a basis of our trust in him – his reliability. The latter is important because we would like to know whether we are able to rely on a recommendation, and the more consistent the semantic distances of previous recommendations the more confident we will be on relying on them in future. For example, if an experience of n frequently follows a recommendation of m then we can be confident that if we receive another recommendation of m then experience with the prospect will likely be at level n.

The range of values for the semantic distance is constrained by the range of trust values used. For the range of five trust values used in Ntropi, the limit for the semantic distance is ± 4, giving the range of $[-4.. + 4]$. Generally, the semantic distance is in the range $-(\#TRUSTVALUE - 1).. + (\#TRUSTVALUE - 1)$. We define a Z type for semantic distance below:

$$SEMDIST : \mathbb{Z}$$

$$\forall s : SEMDIST; x : \mathbb{Z} \bullet (x = \#TRUSTVALUE - 1) \land (s \in \{-x...x\})$$

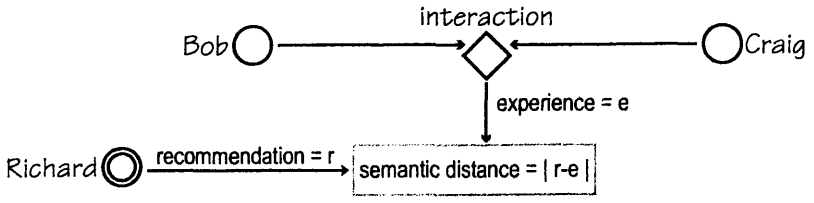


Figure 8.3: Example showing semantic distance result.

8.3 Translating Recommendations

Assume that Bob had received and used Richard's recommendations before, i.e. Bob has had experiences with Richard for context *r1.cServicePayment*. Upon receipt of Richard's recommendation, as shown in Table 8.1, Bob first obtains all past experiences with Richard from his set of experiences with recommenders, *RExperienceSet*. Each member of *RExperienceSet* contains information that includes the context of the past recommendation, the recommended trust level and the semantic distance.

These experiences are then filtered by the context and recommended trust level that are contained in Richard's new recommendation, i.e. the *r1.cServicePayment* context and trust level +I. The typical semantic distance, *tsd* is then obtained from that filtered set, using the median of the set. We assume that in this example, $tsd = +\text{I}$. The translated value is thus obtained by increasing the order of the recommended trust value of $-I$ by 1, yielding the translated value of 0.

The typical semantic distance, *tsd*, is an integer in the range [-4,4]. In the translation process, if the recommender's recommended trust level is *rtv*, then the translated trust value, *ttv*, is a member in the ordered set {-II,-I,0,+I,+II} such that:

$$ord(ttv) = ord(rtv) + tsd$$

The typical semantic distance is only taken from past recommendations for the *same recommended trust level* as in the new recommendation. The reason for this is because in Ntropi, each trust level has different semantics. If we do use all previous experiences (i.e. semantic distances) then we are assuming that the trust level scale is linear, i.e. are 'evenly spaced' between its member values. However, this may not be the case as neighbouring values within the scale may be closer between two levels x and y, than y and z.

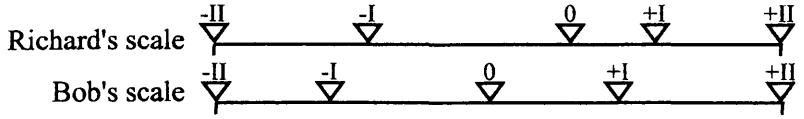


Figure 8.4: Agents' trust scales may not match linearly.

Figure 8.4 illustrates this: Richard's scale is not evenly spaced and his recommended level of 0 translates close to +I, and so does his recommendation of +I, instead of +II, if Bob were to include all of his experiences with Richard (within the *r1.cServicePayment* context).

We are currently uncertain whether the edge cases (-II and +II), which have a limiting effect on

possible trust levels, will be a problem. For example, the scale difference in Figure 8.5 may arise if the recommender has been exposed to a greater range of experiences than the trustee. Future work will look at the significance of this scenario.

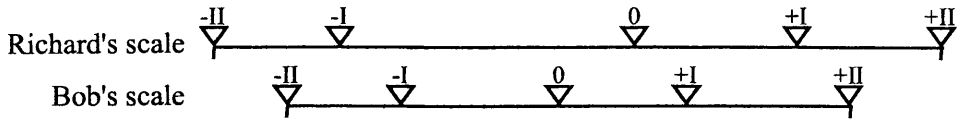


Figure 8.5: Scales of unequal semantic ranges.

The preceding algorithms are presented formally below. Recommendations received are stored in the *RecommendationSet* set whose members are of type *Recommendation*, and there can only be one recommendation from a recommender to a prospect for a context.

Recommendation

$$
\begin{array}{l}
timestamp : DATETIME \\
recommender : AGENTID \\
prospect : AGENTID \\
context : Context \\
trustLevel : TRUSTVALUE
\end{array}
$$

$$RecommendationSet == \mathbb{P}\, Recommendation$$

$$
\begin{array}{l}
\forall r1, r2 : Recommendation \mid r1 \in RecommendationSet \\
\quad r2 \in RecommendationSet \land r1 \neq r2 \\
\quad \Rightarrow \neg\, (r1.recommender = r2.reocmmender \land r1.prospect = r2.prospect \\
\qquad \land\, r1.context = r2.context)
\end{array}
$$

The set of past experiences with a recommender, *RExperienceSet*, contains members of type *RExperienceRec*. The translating is done by calling the function *translateRec*.

RExperienceRec

$$
\begin{array}{l}
timestamp : DATETIME \\
recommender : AGENTID \\
context : Context \\
trustLevel : TRUSTVALUE \\
semDist : SEMDIST
\end{array}
$$

$$RExperienceSet == \mathbb{P}\, RExperienceRec$$

translateRec

$$
\begin{array}{l}
\Xi RExperienceSet \\
rec? : Recommendation \\
result! : TRUSTVALUE \\
d : context \\
SemDists : \mathbb{P}\, SEMDIST \\
\hline
SemDists == \{ \forall rr : ExperienceRec \mid \\
\quad (d.contextID = rr.context.contextID) \land (d.depth = rr.context.depth) \\
\quad \land (rr \in RExperienceSet) \land (rr.context = rec?.context) \\
\quad \land (rr.trustLevel = rec?.trustLevel) \land (rr.recommender = rec?.recommender) \\
\quad \bullet\ rr.semDist \} \\
\\
(SemDists \neq \emptyset) \Rightarrow (result! = adjustLevel(rec?.trustLevel, median(SemDists))) \\
(SemDists = \emptyset) \Rightarrow (result! = rec?.trustLevel)
\end{array}
$$

The function *adjustLevel* in the predicate part of *translateRec* above does the actual 'adjustment' of the recommended level:

$$adjustLevel \mathrel{\widehat=} [t?, result! : TRUSTVALUE;\ n? : \mathbb{Z} \mid \forall t?, n? \bullet ord(result!) = ord(t?) + n?]$$

8.4 Recommender Consistency

Just as with direct context relationships, experience trust for relationships in the recommend context are based on previous experiences with the recommender. A recommender's trustworthiness depends on the consistency of his recommendations. Consistency of the quality of his recommendations forms the basis of a recommender's reliability.

The consistency measurement matches one of McKnight and Chervany's four most prevalent trust-related beliefs: predictability (see §2.3). We argue that consistency of behaviour affects predictability of behaviour in such a way that the more consistent an agent's quality of recommendation, the better we will be able to predict and rely on his recommendations. Predictability is also indicated as one of the functions of reputational information by Hardin (see §2.8.2).

The consistency measurement is based on past semantic distances, from previous experiences with the recommender, for the active context. If the distribution of semantic distances are more spread out, then there is less consistency. The less spread out it is the more consistent the recommender.

Figure 8.6 shows two example distributions of semantic distances in previous experiences with two different recommenders. Recommender A is said to be more consistent than recommender B because a large proportion of his previous recommendations had a semantic distance of -2. Compare this with recommender B where the distribution is more spread out, even though the mean semantic distance is 0. What this means is that whenever we get a recommendation from A for this context and level, then it is very likely that it will match an experience level of -2 in the outcome, because statistically

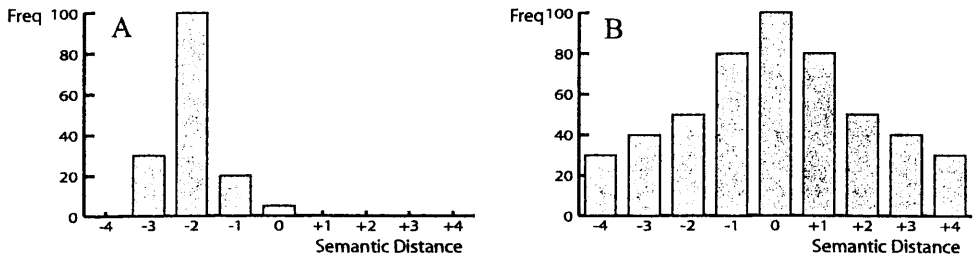


Figure 8.6: Different spreads of semantic distance distribution. A is more consistent than B.

this has been so in the past. This is less certain for B where historically his recommendation of this level has led to many experiences across the range of possible values, thus making the 'predictive power' of his recommendation weaker. Recommender A is said to be more consistent in the quality of his recommendations, hence more trustworthy, than recommender B.

In our model, the consistency is obtained by first finding the semi inter-quartile ranges (SIQR) of the ordered set of semantic distances for the active context, rounded to the nearest integer. Then the lookup table in Table 8.2 is used to convert the SIQR into a trustworthiness level.

Consistency (SIQR)	0	1	2	3	4
Trust level	+II	+I	0	-I	-II

Table 8.2: Consistency-trust level lookup table.

The SIQR of ordered set S is given by:

$$SIQR(S) = \frac{(Q3-Q1)}{2} \quad \text{[Jai91]}$$

where S is a Z *sequence* whose members are of type (n, sd), n representing the position of the member sd in S, and:

$$\exists n : \mathbb{N} \bullet (n, Q1) \in S \Rightarrow n = round(1 + ((\# \operatorname{ran} S - 1) * 0.25))$$
$$\exists n : \mathbb{N} \bullet (n, Q3) \in S \Rightarrow n = round(1 + ((\# \operatorname{ran} S - 1) * 0.75))$$

So if Richard's set of semantic distances is:

$$\{-1, -1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2\}$$

then the SIQR will be:

$$\begin{aligned} SIQR &= (Q3 - Q1)/2 \\ &= (1 - 1)/2 \\ &= 0 \end{aligned}$$

From the consistency lookup table in Table 8.2, this gives Richard a trustworthiness level of +II. The *getConsistency* function contains the preceding algorithm.

getConsistency

$\Xi RExperienceSet$
$forRecommender? : AGENTID$
$forContext? : Context$
$result! : TRUSTVALUE$

$$SD == \{ sd : \mathbb{N} \mid (\exists rr : RExperienceRec \bullet (rr \in RExperienceSet) \land (rr.recommender = forRecommender?) \land (rr.context = forContext?) \land (\mid rr.semDist \mid = sd)) \}$$

$$OSD == \{ SDSeq : seqSD \mid \forall n, x, y : \mathbb{N} \bullet ((n, x) \in SDSeq) \land ((n+1, y) \in SDSeq) \Rightarrow (x \leq y) \bullet (n, sd) \}$$

$$result! = lookupConsistency(siqr(OSD))$$

siqr

$OSD? : \mathbb{P}\,\mathbb{N} \mapsto \mathbb{N}$
$result! : \mathbb{N}$
$q1, q3 : \mathbb{N}$

$$\exists n : \mathbb{N} \bullet (n, q1) \in OSD? \Rightarrow n = round(1 + ((\#\,\mathrm{ran}\, OSD? - 1) * 0.25))$$
$$\exists n : \mathbb{N} \bullet (n, q3) \in OSD? \Rightarrow n = round(1 + ((\#\,\mathrm{ran}\, OSD? - 1) * 0.75))$$

$$result! = (q3 - q1)/2$$

lookupConsistency

$consistency? : \mathbb{N}$
$result! : TRUSTVALUE$
$LookupTable : \mathbb{P}\,\mathbb{N} \mapsto TRUSTVALUE$

$$\exists t : TRUSTVALUE \bullet (consistency?, t) \in LookupTable \Rightarrow result! = t$$

The SIQR is just one approach to finding spread of semantic distances in our model. Other measures of dispersion in the data may be more appropriate for different applications, especially one where the requirement of unbounded, unimodal and symmetrical distribution (for which the SIQR is suitable for[Jai91]) does not exist. The SIQR, however, does not include all data points in the distribution, which may be another consideration when determining an appropriate spread measurement – the

standard deviation, for example, does include all data points.

Furthermore, when converting the SIQR (or whatever the spread measure is) into a trust level, linear conversion need not be assumed, contrary to how the conversion table in Table 8.2 was defined. Table 8.2 is an example of how it can be done. However, one may select different trust values for each SIQR value, depending on the weight one gives to the different SIQR/spread values, such as in Table 8.3.

Consistency (SIQR)	0	1	2	3	4
Trust level	+II	+II	-II	-II	-II

Table 8.3: Alternative consistency-trust level lookup table.

8.5 Recommendation Chains

If a recommender is not known by the recommendation requester, the requester can obtain recommendations about the unknown recommender. There is also the scenario where a recommendation requester may carry out a network search for a particular agent and the received recommendation may be the result of the request being forwarded through a number of intermediary 'recommenders' [Jos02]. In both scenarios, when a recommender recommends another recommender, the result is a *recommendation chain*, two examples of which are illustrated in Figure 8.7, (b) and (c). Recommendation of type (a) is exactly the same as in the previous example with Richard, shown here for comparison. In a recommendation chain each individual path segment carries a different context of trust relationship.

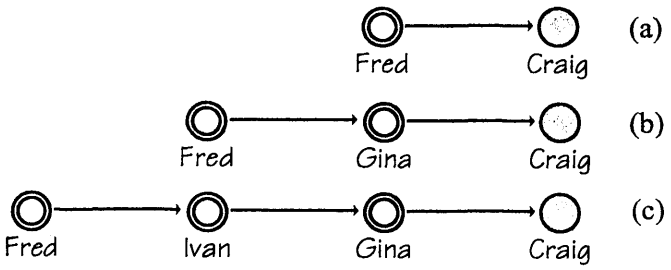


Figure 8.7: Example recommendation chains. Read arrow as "recommends".

In the example chains in Figure 8.7, Bob's relationship with Fred in chain (a) is different to Bob's relationship with Fred in chain (b). While the context for Bob's trust in Fred in (a) is for recommending a direct context prospect, Bob's relationship with Fred in (b) concerns Fred's trustworthiness in recommending another recommender whom in turn is trusted to recommend a direct context prospect. In (a), Fred's incontext has depth index 1 but in (b) his incontext has depth index 2. Different contexts in a recommendation chain is discussed in a little more detail on page 98.

Given that in a recommendation chain with more than one recommender the ultimate recommender in the chain is not known (e.g. Gina in Figure 8.7), it is thus not possible to do any translation as we did with direct recommenders in the previous section (with the example where Richard recommended Craig). Therefore, given a recommendation chain, the recommendation for the direct context prospect must be accepted as is. However it is possible to place restrictions on recommender chains to allow the truster to manage uncertainty in the chain and reduce risk taking if necessary.

This is done with the *chain criteria*, a policy specified for recommenders in an arbitrary chain for a given context.

A recommendation chain can be broken down into three sections (see Figure 8.8). The *heads* of the chain may contain more than one known recommenders, all of which recommends the same first intermediary of the chain. The *intermediaries* are one or more recommenders that are on the path towards the final recommendation for the *direct prospect.* We can represent this with the following Z schema:

Chain

$$heads : \mathbb{P}\, Recommendation$$
$$mid : \mathbb{P}\, Recommendation$$
$$last : Recommendation$$

$$heads \neq \emptyset$$
$$mid \neq \emptyset$$

8.5.1 Chain Criteria

When seeking recommendations or testimonials about another agent, Ntropi attempts to mimic the social process of word-of-mouth. Specifically, an agent seeking recommendations about an unknown prospect will request recommendations from those recommenders that he already knows and trusts. Thus, since a chain's heads are known recommenders, the first stage in determining the acceptance of a recommendation from a recommendation chain is to ensure that the root of the chain is a known recommender. The recommendation requester then specifies a minimum trustworthiness criteria that the first recommender in the chain, or the head of the chain, must meet for the context appropriate to the head's role in the chain. This is the *head criteria.*

If and when the head criteria is met, we can then start considering each intermediate recommender in turn, evaluating each against the requester's minimum trustworthiness criteria for intermediaries, the *intermediary criteria.* The intermediary criteria is essential because recommended recommenders may not have the minimum level of trust sufficient for the truster to have confidence in their recommendations. For example, in Figure 8.7(b), Alice may only accept Gina's recommendation only if Fred says Gina is a 'very trustworthy' recommender, and no less.

In Figure 8.7(c), the head criteria will apply to Fred, the head of the chain, and the intermediary criteria will apply to Ivan and Gina, the intermediary recommenders. The recommendation chain criteria are to be tested against the incontext of the recommender being evaluated. They are specified in the truster's Ntropi policy, parameters for each criterion detailed in Table 8.4.

The head *ConsistencyTrust* parameter is the consistency trust measurement returned by algorithm *getConsistency* on page 171. *headCount* indicates the minimum number of known recommenders required to be at the head of the chain. In other words, how many known recommenders are required to recommend the first unknown intermediary recommender in the chain. Figure 8.8 is an example chain with three known recommenders at the head of the chain. This models the social phenomenon

Criteria	Parameter
Head	*headConsTrust* $\geqslant t$. This is the head's consistency based trustworthiness.
	headCount $\geqslant s$. This is the minimum required number of heads on the chain, specified as number of stable phase relationships.
Intermediary	*intReputation* $\geqslant rr$. This is the minimum reputation of an intermediary as assigned by the heads or a prior intermediary.

Table 8.4: Chain criteria and their parameters.

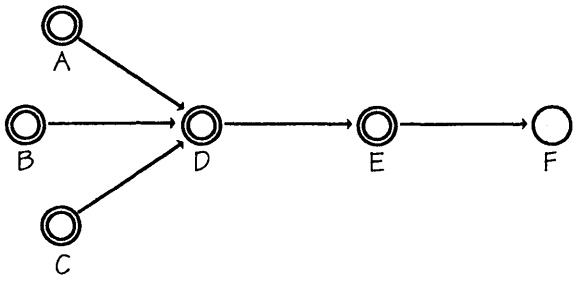


Figure 8.8: A chain with three 'heads' (A, B and C), two intermediaries (D and E) and one direct prospect. Read arrow as "recommends".

where confidence is raised through an increase in number of trusted referrals.

This parameter is defined in terms of number of stable phase recommenders. For example, if *headCount* = *Stable* × 2 then two known heads are required and they both have to be in the stable phase. To allow flexibility in meeting the head criteria, a phase equivalence policy can be asserted for each context in the form:

$$Fragile \times m = Stable \times n$$

This allows Fragile recommenders to be used to meet the head criteria of a chain. For example, if Bob asserts that for the context *cServicePayment*:

$$Fragile \times 3 = Stable \times 1$$

then, in the example in Figure 8.8, if A is a required stable phase recommender, she can be replaced with three other Fragile recommenders. Phase equivalence is specified in the Ntropi policy and as such can be asserted using the policy schemas on page 143. The following defines the phase equivalence policy structure, which maps the number of Fragile relationships required to the number of equivalent stable phase relationships.

$$PhaseEquivRec ::= \mathbb{N}_1 \nrightarrow \mathbb{N}_1$$

The intermediary *Reputation* and *Reliability* parameter is calculated according to the *getReputation*

and *getConsistency* functions respectively. These were defined on pages 184 and 171. Figure 8.9 illustrates various recommendation chains and shows where in the chain the different criteria apply.

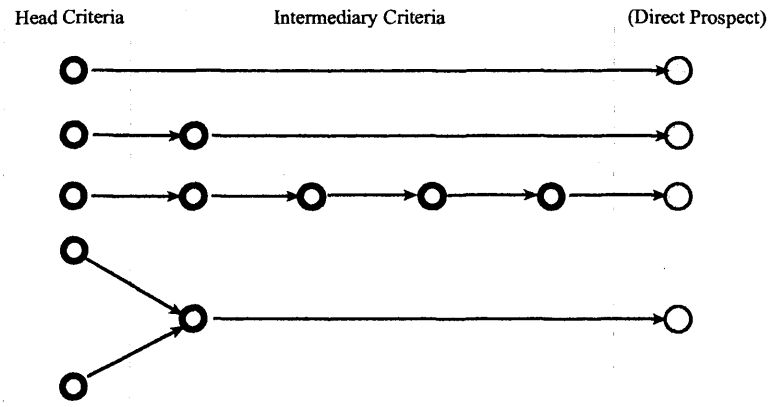


Figure 8.9: Various recommendation chains and where the different chain criteria apply. Read arrow as "recommends".

Chain criteria for each context is stored in *ChainCriteriaRec* and the collection of criteria is stored in *ChainCriteriaSet*. The functions *addChainCriteria* and *getChainCriteria* facilitate assertion and retrieval of these chain criteria.

ChainCriteriaRec

$$
\begin{array}{l}
headConsTrust : TRUSTVALUE \\
headCount : \mathbb{N} \\
intReputation : TRUSTVALUE \\
maxRecDepth : \mathbb{N}
\end{array}
$$

$$ChainCriteriaSet == CONTEXTID \rightarrow ChainCriteriaRec$$

addChainCriteria

$$
\begin{array}{l}
\Delta ChainCriteriaSet \\
forContext? : CONTEXTID \\
newCriteriaRec? : ChainCriteriaRec \\
\hline
ChainCriteriaSet' = ChainCriteriaSet \oplus \{forContext? \mapsto newCriteriaRec?\}
\end{array}
$$

getChainCriteria

$\Xi ChainCriteriaSet$

$forContext? : CONTEXTID$

$result! : ChainCriteriaRec$

$\exists rec : ChainCriteriaRec \bullet (forContext? \mapsto rec) \in ChainCriteriaSet \Rightarrow result! = rec$

The following illustrates how recommendation chains are checked against the chain criteria. We assume that Bob defined his external trust management policy as in Table 8.6. Bob receives another recommendation about Craig for the context *cServicePayment* and trust level +I, from a recommender called Gina. Because Bob does not know Gina, he asks Fred and Helen whether they know Gina and could tell Bob about Gina's trustworthiness with respect to giving *cServicePayment* recommendations. They both reply with a recommendation about Gina with outcontext *r1.cServicePayment* and trust levels +I. The three recommendations are shown in Table 8.5.

Recommender	Prospect	Context	Depth index	Trust level
Gina	Craig	cServicePayment	0	+I
Fred	Gina	cServicepayment	1	+I
Helen	Gina	cServicePayment	1	+I

Table 8.5: Example recommendations for a chain.

From these recommendations Bob will build the recommendation chain, by connecting the agents in the recommendations and identifying their relevant incontexts. The resulting chain is shown in Figure 8.10.

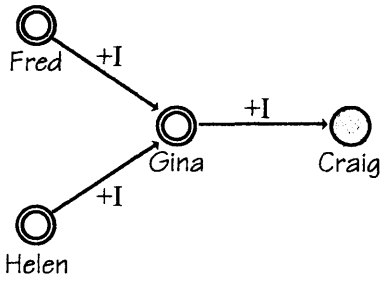


Figure 8.10: Example chain for context *cServicePayment*. Recommended trust levels shown next to arrow.

Chain criteria for context: *cServicePayment*	
Criteria	Parameter
Head	*headConsTrust* $\geqslant$ +I
	headCount $\geqslant 2$
Intermediary	*intReputation* $\geqslant$ +I

Table 8.6: Example chain criteria.

Table 8.7 shows the recommenders from the previous example, their relevant incontexts and the relevant applicable criteria from the example policy in Table 8.6.

Let's assume that Fred and Helen are Bob's stable phase recommenders in incontext *r2.cServicePayment*

Recommender	Incontext	Head Criteria Consistency Trust	Intermediary Criteria Reputation	Intermediary Criteria Reliability
Fred	r2.cServicePayment	+I	N/A	N/A
Helen	r2.cServicePayment	+I	N/A	N/A
Gina	r1.cServicePayment	N/A	+I	+I

Table 8.7: Example recommendation chain unpacked.

and their consistency trustworthiness values are +I and +II respectively. The head criteria, which applies to Fred and Helen, says that at least two stable phase recommenders are required and they should have at least a consistency trust value of +I. Since Fred and Helen meet these criteria, Bob moves on to the next recommender in the chain, Gina.

For Gina, Bob must now check her reputation against the intermediary criteria. The relevant incontext for Gina is *r1.cServicePayment*. The requirement is for Gina's reputation in this context to be at least +I and this should have a reliability of at least +I. By combining the recommendations from Fred and Helen (combination of recommendations will be looked at in the next section). Bob gets Gina's reputation, which is +I. Furthermore, the reliability measure of this is +I. Thus Gina passes the test. Since Gina is the final recommender in the chain (depth index = 1), the evaluation terminates successfully and the recommendation is accepted.

This gives the final values for this recommendation chain as follows: Recommended trust value = +I (from Gina) and Reliability = +I. The reliability of this recommendation chain is taken from the median of the reliabilities of the chain's heads.

A possible weakness of the approach taken in Ntropi, where all chain heads must be known, is that it will not be possible to accept recommendations from chains with unknown heads, even if the requester is willing to use those recommendations. An example where unknown recommenders may be useful is when the alternative is to have no recommendation at all. This situation was discussed in the social science chapter where the example of asking for directions on the street demonstrated that at times one may use advice from a stranger, i.e. when nothing is known about the recommending agent. This is particularly true in situations where possession of any information is better than no information, and, at the same time, there is belief in the benevolence of the recommender as well as low perceived risk. This shows us again that trust decision-making involves factors beyond just evaluating the trustworthiness of prospects, which, in this work, is seen as the responsibility of the External Trust Management component because it is application specific. Future work will address how Ntropi can assist External Trust Management modules in this respect.

8.5.2 Chain Length

In general, the shorter the recommendation chain, the lower the uncertainty in the recommendation and the better the quality of the recommendation. For the example chain in Figure 8.10, the best case would be for Bob to know Craig directly (in which case he would not need the recommendation at all), next would be for Bob to already know and used recommendations from Gina (one recommender away from prospect) and next best after that would be for Bob to know and have used recommendations from Fred or Helen (two recommenders away from prospect). The worst case would be to not know any of the entities in the chain.

Thus, one's task when given a recommendation chain would be to try and shorten the chain as much as possible and to look for the recommender or prospect closest to the direct prospect at the end of the chain. In the given example, Bob will start from Craig and then work his way towards Fred to find the first agent he has had experiences with.

Trimming Chains

If Craig is known, then Bob 'trims' the recommendation at Craig, removing Gina and Fred from the chain. Bob then evaluates Craig's trustworthiness as in the previous chapter, with respect to context *d.cServicePayment*. If Craig is not known to Bob then, moving up the chain by one agent, Bob checks to see if Gina is known. If she is, then the chain is trimmed at Gina, removing Fred from the chain. If Gina is not known then Bob moves up the chain again now arriving at Fred, and if known, evaluates the recommendation at Fred.

Maximum Chain Length

It is also possible to ignore chains that are longer than a maximum allowable length. For each context, the maximum allowable chain length can be set with its *maxRecDepth* parameter. This tells Ntropi to ignore recommendation chains with more than this number of recommenders in the evaluation of reputation. In other words, it tells Ntropi to ignore recommendations whose context depth index is greater than *MaxRecDepth*. This method is similar to that used in various other trust models, e.g. PGP and X.509 (see §4.6.1). For example, if Bob wanted to restrict recommendations in the *cServicePayment* context to only direct recommendations only, he can set the *maxRecDepth* parameter for the cServicePayment context to 1. The result of this is that only contexts of $r1.cServicePayment$ will be considered.

Below we provide the Z schema for validating recommendation chains against chain criteria. The *validateChains* accepts a set of recommendation chains, checks them against the chain criteria and returns the set of chains that meet the criteria. The function first obtains the chain criteria by calling *getChainCriteria*, and then, for each chain, validates each criteria parameter by calling the appropriate functions: *validateHead* to validate the heads of the chain and *validateMiddle* to validate the intermediate recommenders. The chain is also checked against the maximum allowable chain length against the *maxRecDepth* parameter. The chain length is obtained by obtaining the number of intermediary recommenders ($\#c.middle$) and adding 2 to it, to represent the head and the final (direct) recommender.

validateChains

$$
\begin{array}{l}
Chains? : \mathbb{P}\, Chain;\ forContext? : CONTEXTID \\
result! : \mathbb{P}\, Chain \\
crit : ChainCriteriaRec \\
\hline
crit = getChainCriteria(forContext?) \\
result! = \{\, c : Chain \mid (c \in Chains?) \\
\quad \land\ (validHead(c, forContext?)) \land (validMiddle(c, forContext?)) \\
\quad \land\ (\#c.middle + 2) \leqslant crit.maxRecDepth) \,\}
\end{array}
$$

validHead evaluates the set of heads of a given chain, by the following algorithm:

1. Get the chain criteria.
2. Get the phase equivalence record, which is a $\mathbb{N}_1 \mapsto \mathbb{N}_1$ tuple, and calculate *valuePerFragile*, which is the value of one Fragile relationship as a fraction of one Stable relationship.
3. *valuePerFragile* is then multiplied by the number of recommenders in the head that the truster has a Fragile relationship with.
4. This product is then added to the number of recommenders in the head that the truster has a Stable relationship with.
5. The head portion of this chain is valid if the final sum above equals or exceeds the *headCount* criteria.

validHead

$$
\begin{array}{l}
c? : Chain;\ forContext? : CONTEXTID \\
result! : TRUE \mid FALSE \\
crit : ChainCriteriaRec;\ valuePerFragile : \mathbb{R};\ d : CONTEXT \\
\hline
crit = getChainCriteria(forContext?) \\
valuePerFragile = second(getPolicy(npPhEquiv, forContext))/ \\
\quad first(getPolicy(npPhEquiv, forContext)) \\
Fragiles = \{ r : Recommendation \mid r \in c?.heads \\
\quad \land\ d.contextID = r.context.contextID \\
\quad \land\ d.depth = r.context.depth + 1 \\
\quad \land\ getPhase(r.recommender, d) = phFragile \} \\
\\
\#(c?.heads \setminus Fragiles) + rounddown(\#Fragiles * valuePerFragile) \geqslant crit.headCount \\
\quad \Rightarrow result! = TRUE
\end{array}
$$

validMiddle checks the validity of the intermediary recommenders of a chain by the following algorithms:

1. Get chain criteria.
2. Ensure that the reputation, as recommended by the heads of the chain, of the first recommender in the intermediary sub-chain, meets the minimum criteria, and is unfamiliar. The head of this subchain is identified by the depth of its recommendation context, which should equal the length of the intermediary subchain.
3. Each intermediary recommender must have been recommended by immediate predecessor by at least the minimum level defined in the criteria. Since all recommendations in the intermediary portion are about other recommenders within the intermediary sub-chain, this criteria can

be checked by simply checking that all recommendations recommended a minimum of the trust level in the criteria.

validMiddle

$c? : Chain$; $forContext? : CONTEXTID$
$result! : TRUE \mid FALSE$
$crit : ChainCriteriaRec$; $d : CONTEXT$

$crit = getChainCriteria(forContext?)$

$\exists r : Recommendation \mid r.context.depth = \#c.middle \wedge$
$d.contextID = r.context.contextID \wedge$
$d.depth = r.context.depth + 1 \wedge$
$getReputation(c?.heads, r.recommender, d) \geqslant crit.intReputation \wedge$
$getPhase(r.recommender, d) = phUnfamiliar$

$\forall r : Recommendation \mid r \in c?.middle \wedge$
$ord(r.trustLevel) \geqslant ord(crit.intReputation) \wedge$
$d.contextID = r.context.contextID \wedge$
$d.depth = r.context.depth + 1 \wedge$
$getPhase(r.recommender, d) = phUnfamiliar$
$\Rightarrow result! = TRUE$

8.6 Combining Recommendations into Reputation

A truster may have more than one recommendation for a prospect and wish to aggregate the opinions made in these recommendations to arrive at a general *reputation* level for the prospect. A simple method could be to obtain just the average recommendation from the received recommendations. However, we may decide that some recommenders are more trustworthy, then their recommendations carry more weight. This mirrors real world scenarios where we trust the opinions of certain agents more than others. In this model we allow weights to be placed on recommendations based on the level of trust of each recommender, i.e. by weighting according to the recommender's individual consistency trust levels. To facilitate this, the truster makes use of the Consistency Weight Table.

For direct recommendations (chains with only one recommender), any recommender in the untrusted phase or with an unknown trust level is discarded as we only want to consider recommendations from trusted agents. Each recommendation is then translated using the *translateRec* function as shown on page 169. In this initial step chains with depth > 1 are checked against the chain criteria – those not meeting the requirements are discarded.

All recommendations are next weighted according to the consistency of their recommenders, or chain reliability. The weights, called *conweights*, as shown in Table 8.8, represent the ratio of 'strength of opinion' assigned to recommenders with different levels of trust. The idea is that the

more trustworthy (i.e. consistent) the recommender, the more reliable his recommendation, and subsequently, the greater weight his judgement. These weights can be defined per context and is customisable by the truster. By allowing different weights to be assigned to different trust levels, Ntropi allows non-linear ratios to be modeled.

We shall observe here that the weighted trust level approach provides customisation and flexibility in weighting recommendations based on the trust levels of their recommenders, it also adds to the user's list of tasks to perform, namely that he must be able to define, and adjust, if required, the *conweights* for each application that uses this model. In reality, this is not a very satisfactory situation, and as such, will require additional help from the application itself in terms of either hardwiring the weighting based on well known properties of agents in the application domain, or employ some form of learning algorithm that can dynamically update the weights based on experience. User interface issues and dynamic updating of weights and policies is a subject of future research.

Consistency/Reliability	-II	-I	0	+I	+II
Weight	$c1$	$c2$	$c3$	$c4$	$c5$

Table 8.8: Consistency trust weights, *conweights*.

conweights are defined in the *Conweights* set, here defined as a function mapping a context to its corresponding table.

$$Conweights == CONTEXTID \pfun (TRUSTLEVEL \fun \mathbb{N})$$

The function *addConweight* defines or updates conweights for a given context.

addConweight

$\Delta Conweights$

$forContext? : CONTEXTID$

$newConweights? : TRUSTLEVEL \fun \mathbb{N}$

$Conweights' = Conweights \oplus (forContext?, newConweights?)$

Furthermore, because there is generally a higher level of uncertainty for relationships in the Fragile phase (as the relationship is too short to make any sense of consistency), recommenders in the Fragile phase are treated as having a trust level of 0. Thus, according to Table 8.8, any recommendations from Fragile recommenders will be assigned a weight of $c3$[1].

For recommendation chains, the consistency trust for the recommendation as a whole is taken from the heads of the chain (see previous section).

Sum the *conweights* associated with the recommendations for that level for each trust level that has been recommended. The combined reputation level for the prospect is the trust level with highest

[1]Here we have introduced an element of subjectivity into the model – our own disposition towards Fragile phased relationships. It would be trivial to model this as a variable, however we leave it as is as an illustration of how the design of a trust model can itself be influenced by the trusting dispositions of its designers.

sum of weights. In the case of a 'draw' between more than one level, the General Trust Disposition for the active context is used to select the final level from the list of candidate levels.

Given that the goal of our recommendation combination algorithm is to be able to use recommendations from the most trustworthy recommenders, the proposed approach is, we believe, a reasonable solution to the problem. However, a few caveats.

Firstly, since our algorithm results in the selection of recommendations with the highest weightings, we will potentially be ignoring other recommendations that also originated from trustworthy recommenders, albeit from those with lower comparative trustworthiness levels. The approach we have taken handles this issue to some extent, in that, given a sufficiently high number of recommendations (for the same trust value) from lower trust recommenders, their recommended trust values may still be the 'winning' value because their sum-of-weights will outweigh the recommenders with higher trust but with a lower population within the local set of recommendations. However, it can then be argued that we will then be using less trustworthy recommenders at the expense of ignoring those from a higher trusted source, simply because there were more of the lower trust recommenders giving recommendations than the higher trust ones.

Neither extreme is satisfactory, and a better algorithm would be one where a *new* trust value is produced by the reputation/combination algorithm based on the recommendations received from all the trusted recommendations, from the whole range of trust levels. Some work towards this have been carried out in the research community, such as some of those surveyed in Chapter 4, but the algorithms presented lack grounding in any real world examples of how recommendations are combined, and hence are ad-hoc in nature. Thus, our motivation was to offer an alternative approach based on understanding of trust dynamics in the social sciences. However, it is acknowledged that much work is still required in this specific recommendation combination problem.

We illustrate the recommendation combination process above with the following example. Assume that Bob also received recommendations from Saul, Tracy and Ulrich, in addition to Richard's recommendation. In addition he also has two recommendation chains, as shown in Figure 8.11. Tracy is untrusted so her recommendation is discarded. The table shows each recommender's recommendation (translated, for known recommenders) and their consistency trust. Bob's conweights are shown in Table 8.9.

Consistency Trust	-II	-I	0	+I	+II
Weight	0	0	1	2	5

Table 8.9: Bob's example conweights for context $r1.cServicePayment$.

Rec'der	Phase	(T'lated) Recommendation	Consistency	conweight
Richard	Stable	0	0	1
Saul	Stable	+I	+II	5
Ulrich	Fragile	0	0	1
~~Tracy~~	~~Untrusted~~	~~+II~~	~~n/a~~	~~n/a~~
Chain1	N/A	+I	+I	2
Chain2	N/A	0	+I	2

Table 8.10: Example recommendations to Bob about Craig in the context *cServicePayment*.

We can see that two different levels were recommended: level 0 by Richard, Ulrich and Chain2, and

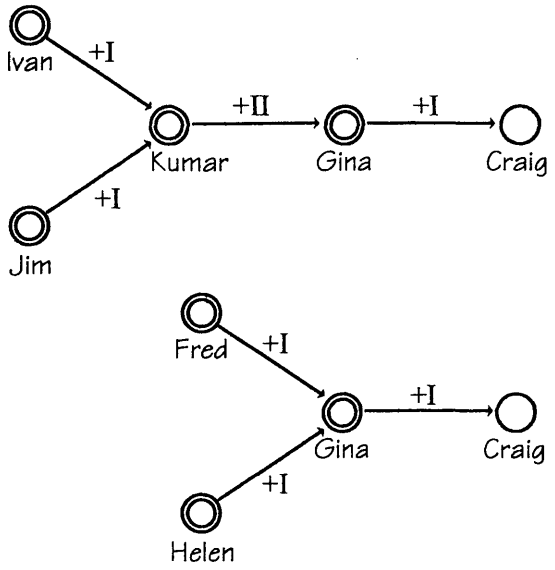


Figure 8.11: Recommendation chains received by Bob about Craig for context *cServicePayment*.

level +I by Saul and Chain1. Total conweights for the recommended level 0 is 4 (1+1+2) and for level +I is 7 (5+2). This means that the final reputation value for Craig for context *cServicePayment* is +I as it has the highest sum of conweights.

The function *getReputation* embodies the main combination algorithm. The function returns a tuple (*rep*, *rel*) where *rep* it the combined trust level representing the prospect's reputation level, and *rel* is a value indicating the reliability of this reputation. It follows the following algorithm:

1. Remove direct recommendations not directed at the prospect from the set of recommendations.
2. Separate direct recommendations with known recommendations (*DirectRecs*) from recommendation that are part of a chain (*ChainRecs*).
3. Create two sets, one containing valid chains (*ValidChains*) and the other containing valid direct recommendations (*ValidDirectRecs*).
4. From the two valid sets above, create two sets containing the recommended trust values and the weights attached to them, and then combine them into a single set (*cw*).
5. Select the final reputation value by calling *combineSelect*(*cw*).
6. Calculate the final reliability value by calling *combinedReliability*(*validDirectRecs*, *ValidChains*). Reputation reliability is simply the median of the trustworthiness level of all the known heads of chains plus known direct recommenders.
7. Return result tuple.

makeChains takes a set of recommendations, a direct prospect and context and returns a set containing all recommendation chains leading to the direct recommendation about the prospect for the given context. The result excludes direct recommendations from known direct recommenders.

makeChains

$$
\begin{array}{l}
Recommendations? : \mathbb{P}\, Recommendation \\
forContext? : CONTEXTID;\ forProspect? : AGENTID \\
result! : \mathbb{P}\, Chain \\
\hline
result! = \{\, c : Chain \mid c.last.contextID = forContext? \\
\quad \wedge\ c.last.prospect = forProspect? \wedge c.last.context.depth = 0 \\
\quad \wedge\ hasMiddle(c) \wedge hasHeads(c)\,\}
\end{array}
$$

getReputation

$$
\begin{array}{l}
Recommendations? : \mathbb{P}\, Recommendation; \\
forProspect? : AGENTID;\ forContext? : CONTEXTID \\
result! : TRUSTVALUE \times TRUSTVALUE \\
WeightedChain, WeightedDirectRecs : TRUSTVALUE \rightarrowtail \mathbb{N} \\
d : Context \\
\hline
\forall r : Recommendation \bullet (r.context.depth = 0) \wedge (r.prospect \neq forProspect?) \\
\quad \Rightarrow r \notin Recommendations? \\
\\
ChainRecs = \{\, r : Recommendation \mid (r \in Recommendations?) \wedge \\
\quad ((r.context.depth = 0) \wedge (d.contextID = r.context.contextID) \wedge \\
\quad (d.depth = r.context.depth + 1) \wedge (getPhase(r.recommender, d) = phUnfamiliar)) \\
\quad \vee \\
\quad (r.context.depth > 0)\,\} \\
DirectRecs = Recommendations \setminus ChainRecs \\
\\
ValidChains = validateChains(\\
\quad makeChains(ChainRecs, forProspect?, forContext?), forContext?) \\
ValidDirectRecs = validateDirectRecs(DirectRecs, forProspect?, forContext?) \\
\\
WeightedChains = weighChains(ValidChains) \\
WeightedDirectRecs = weighDirectRecs(ValidDirectRecs) \\
\\
cw = WeightedChains \bigcup WeightedDirectRecs \\
\\
result! = (combineSelect(cw, forContext?), \\
\quad combinedReliability(validDirectRecs, ValidChains))
\end{array}
$$

The *hasMiddle* and *hasHeads* schemas validate the heads and intermediary components of a chain and return a TRUE or FALSE.

hasHeads

$c? : Chain$

$result! : TRUE \mid FALSE$

$\forall r : Recommendation \mid r \in c?.heads \bullet \quad r.context.depth = \#c?.middle + 1$

$\land \forall m : Recommendation \mid r \in c?.middle \bullet$

$(m.recommender = h.prospect) \Rightarrow (m.context.depth = \#c?.middle)$

$result! = TRUE$

hasMiddle

$c? : Chain$

$result! : TRUE \mid FALSE$

$(\exists r : Recommendation \mid r \in c?.middle \bullet$

$(r.context.depth = 1) \land (r.prospect = c?.last.recommender))$

$\land$

$(\forall r : Recommendation \mid (r \in c?.middle) \land (r.context.depth > 1) \bullet$

$\exists r2 : Recommendation \mid r \in c?.middle \bullet$

$r.prospect = r2.recommender \land$

$r.context.depth = r2.context.depth + 1$

$\land \forall r2 : Recommendation \mid (r \in c?.middle) \land (r \neq r2) \bullet$

$r.context.depth \neq r2.context.depth \land$

$r.recommender \neq r2.recommender$

$\land \exists r2 : Recommendation \mid r2 \in (c?.head \bigcup c?.middle) \bullet$

$r2.prospect = r.recommender)$

$\Rightarrow result! = TRUE$

validateDirectRecs removes any invalid direct recommendations from a given set of direct recommendations and returns the set of valid recommendations. Valid recommendations are those with known and not untrusted recommenders.

validateDirectRecs

$Recommendations? : \mathbb{P}\, Recommendation$

$forProspect? : AGENTID; \; forContext? : CONTEXTID$

$result! : \mathbb{P}\, Recommendation$

$result! = \{ r : Recommendation \mid r.prospect = forProspect?$

$\land \; r.context.contextID = forContext?$

$\land \; getPhase(r.recommender, r.context) \neq phUntrusted \}$

weighDirectRecs and *weighChains* take a set of direct recommendations or chains respectively and each returns a set of tuples of type (t, w) where t is a recommended trust level and w is the weight attached to it, depending on the recommender who recommended that trust level.

weighDirectRecs

$$
\begin{array}{l}
Recommendations? : \mathbb{P}\, Recommendation \\
result! : TRUSTVALUE \rightarrowtail \mathbb{N} \\
incontext : Context \\
TW : \mathbb{P}(Recommendation \times TRUSTVALUE) \times \mathbb{N} \\
\hline
\mathrm{dom}\, TW = Recommendations? \\
TW = \{ (rec \mapsto weight) \mapsto tl) \mid \\
\quad tl = translateRec(rec) \land \\
\quad incontext.contextID = rec.context.contextID \land \\
\quad incontext.depth = rec.context.depth + 1 \land \\
\quad getPhase(rec.recommender, incontext) = phFragile \\
\qquad \Rightarrow weight = getConWeight(0) \\
\quad getPhase(rec.recommender, incontext) \neq phFragile \\
\qquad \Rightarrow weight = getConWeight(getConsistency(rec.recommender, incontext)) \} \\
result! = \{ item \mapsto totweight \mid totweight = sumweights(\mathrm{dom}(TW \rhd \{ item \})) \}
\end{array}
$$

weighChains

$$
\begin{array}{l}
Chains? : \mathbb{P}\, Chain \\
result! : TRUSTVALUE \rightarrowtail \mathbb{N} \\
TW : \mathbb{P}(Chain \times TRUSTVALUE) \times \mathbb{N} \\
\hline
\mathrm{dom}\, TW = Chains? \\
TW = \{ (chain \mapsto weight) \mapsto tl) \mid \\
\quad tl = chain.last.trustLevel \land \\
\quad weight = getConWeight(combinedHeadCons(chain.heads)) \} \\
result! = \{ item \mapsto totweight \mid totweight = sumweights(dom(TW \rhd \{ item \})) \}
\end{array}
$$

The *sumweights* function takes a relation with a generic domain and range of type $\mathbb{N}$ and returns the sum of the range in the relation.

sumweights[*DOM*]

$$
\begin{array}{l}
X? : \mathbb{P}\, DOM \times \mathbb{N} \\
result! : \mathbb{N} \\
\hline
S == \mathrm{seq}\, X? \\
\#X? = 1 \Rightarrow result! = \mathrm{second}(\mathrm{head}\, S) \\
\#X? > 1 \Rightarrow result! = \mathrm{second}(\mathrm{head}\, S) + sumweights(\mathrm{ran}\, \mathrm{tail}\, S)
\end{array}
$$

combinedHeadCons simply returns the median of all the consistency trust levels for a the recommenders in the given set of recommendations.

combinedHeadCons

$$
\begin{array}{l}
Recommendations? : \mathbb{P}\, Recommendation \\
result! : TRUSTVALUE \\
incontext : Context \\
RecCons : \mathbb{P}\, Recommendation \times TRUSTVALUE \\
\hline
RecCons = \{ r : Recommendation;\ t : TRUSTVALUE \mid \\
\quad r \in Recommendations? \land \\
\quad incontext.contextID = r.context.contextID \land \\
\quad incontext.depth = r.context.depth + 1 \land \\
\quad t = getConsistency(r.recommender, incontext) \bullet t \} \\
result! = median(RecCons)
\end{array}
$$

The schema *combineSelect* combines the weighted trust values from the direct recommendations and chains and selects the final reputation level, which is the level with the highest total weight.

combineSelect

$$
\begin{array}{l}
WeightedLevels? : TRUSTVALUE \pfun \mathbb{N} \\
forContext? : Contextresult! : TRUSTVALUE \\
\hline
SummedWeights = \{ rtl : TRUSTVALUE;\ totweight : \mathbb{N} \mid \\
\quad S = WeightedLevels \rhd \{rtl\} \land \\
\quad totweight = sumweights(S) \} \\
if\ \#\{SummedWeights \rhd \{\max \operatorname{ran} SummedWeights\}\} = 1 \\
then\ \exists t : TRUSVALUE;\ s : \mathbb{N} \mid t \mapsto s \in SummedWeights \bullet \\
\quad s = \max \operatorname{ran} SummedWeights \land result! = t \\
else\ result! = applyGTD(SummedWeights, forContext?)
\end{array}
$$

combinedReliability calculates the combined trust levels of all direct recommendations and chains together by returning the median of those combined recommendations.

combinedReliability

$DRecs? : \mathbb{P}$; $Chains? : \mathbb{P}\, Chain$
$result! : TRUSTVALUE$
$dTrustVals : Recs? \rightarrowtail TRUSTVALUE$
$cTrustValues : Chains? \rightarrowtail TRUSTVALUE$

$\forall r : Recommendation;\ t : TRUSTVALUE \mid r \mapsto t \in dTrustVals \Rightarrow t = getConsistency(r)$
$\forall c : Recommendation;\ t : TRUSTVALUE \mid c \mapsto t \in cTrustVals \Rightarrow c = combinedHeadCons(c)$
$result! = median(\text{ran}\, dTrustVals \bigcup \text{ran}\, cTrustVals)$

The functions *getConweight* looks up the weight for the given trustworthiness value.

getConweight

$\Xi Conweights$
$forContext? : CONTEXTID$
$tlevel? : TRUSTVALUE$
$w, result! : \mathbb{N}$

$\exists (tlevel?, w) \in Conweights(forContext?) \Rightarrow result! = w$

The *applyGTD* function is called when there is more than one *max* value in the final summed weighted trust levels set, which necessitates a reference to the truster's General Trust Disposition (GTD) to select the appropriate one from the candidates.

applyGTD

$WeightedSet? : \mathbb{P}(TRUSTVALUE \rightarrowtail \mathbb{N})$
$forContext? : CONTEXTID$
$result! : TRUSTVALUE$
$SetofMaxes : \mathbb{P}(TRUSTVALUE \rightarrowtail \mathbb{N})$
$gtd : POLICY$

$SetofMaxes == \{ (rtl, w) \mid$
$\quad ((rtl, w) \in WeightedSet?) \land \neg (\exists (rtl2, w2) \in WeightedSet? \bullet (w2 > w)) \}$

$gtd = getPolicy(npGenTrustDisp, forContext?)$

$gtd = gtdRiskAverse \Rightarrow result! = min(\text{dom}\, WeightedSet?)$
$gtd = gtdRiskSelective \Rightarrow result! = max(\text{dom}\, WeightedSet?)$
$gtd = gtdMedian \Rightarrow result! = median(\text{dom}\, WeightedSet?)$

8.7 Feedback and Phase Updating

After an interaction, the truster will be in a position to carry out any required updating of his set of previous experiences, as well as checking and updating phases of trust relationships. Note that whereas only a single trust relationship is concerned when only using experience trust, recommended trust updating involves at least two relationships, i.e. the truster will also have to update his relationship with the recommender, in addition to his relationship with the trustee in the direct context. Updating of relationships for the direct context have been covered in Chapter 7 so we will now look at the updating of recommender relationships. Updating after an interaction involving recommendations can be broken down into the following tasks:

1. Recording experiences had with recommenders (semantic distances).
2. Checking whether any recommenders require phase transitions.

8.7.1 Recording Experiences

An experience with a recommender is the semantic distance between the recommended trust value from that recommender and the actual experience had with the prospect in the recommendation. After receiving the experience value of an interaction had with a trustee, this experience value is compared with that recommended by the trustee's recommenders. The difference between the experience and recommendation, i.e. the semantic distance, is then recorded in *RExperienceSet*.

Continuing our running example, Bob interacts with Craig after using the recommendations in Table 8.6. Assuming that Craig was given an experience level of 0, the example in Table 8.7.1 shows the resulting semantic distances for each direct recommender (those with incontext depth index 1).

Rec'der	Recommendation	Actual Experience	Semantic Distance
Richard	-I	0	+1
Saul	0	0	0
Ulrich	+I	0	-1
Gina	+I	0	-1

Table 8.11: Example resulting semantic distances for direct recommenders after an interaction.

For recommendation chains, the experience from a recommender in the chain is used to evaluate the recommender's recommender, propagating the experience from the direct recommender right back to the heads of the chain. Thus the semantic distance resulting from a recommendation from a recommender with outcontext depth index n becomes the experience for the recommender with outcontext $n + 1$ up along the chain. However, because the experience must be of type TRUSTVALUE for semantic distance calculations to be carried out for recommender experiences, we must translate semantic distances to trust levels before propagating them up the chain. For this we use a simple linear conversion from absolute semantic distance to trust level, as shown in Table 8.12.

Semantic Distance	4	3	2	1	0
Trust level	-II	-I	0	+I	+II

Table 8.12: Linear conversion from absolute semantic distance to trust level.

The pairs of conversion values in each column in Table 8.12 have been arbitrarily assigned in this instance – the conversion need not be linear, or have any specific distribution. Actual implementations may have their own requirements as to which trust values to be converted from a specific semantic distance. We have also made the assumption here that the absolute value of each semantic distance is of importance, i.e. a measure of the actual distance rather than the complete vector. Again, this may not be the right assumption for certain applications where perhaps the sign of the semantic distance is also important. For example, less trust may be placed on those recommenders that tend to 'over-recommend' than 'under-recommend', as shown in Table 8.13

Semantic Distance	-4	-3	-2	-1	0	1	2	3	4
Trust level	-I	0	+I	+I	+ II	0	-I	-II	-II

Table 8.13: Alternative conversion, placing less trust in recommenders who over-recommend.

Let us take Chain1 from Figure 8.11 as an example. The semantic distance between our experience with Craig and Gina's recommendation is -1, as shown in Table 8.7.1. Using the conversion table in Table 8.12, we convert the absolute value of that resulting semantic distance, which is 1, into a trust level of +I. This level of +I is now taken as our experience with Gina. We now use this to get the semantic distance between this experience with Gina and Kumar's recommendation, giving Kumar's recommendation a semantic distance of -1. Converting this again we get +I as the experience with Kumar, and this is used for evaluating the semantic distance of Ivan's and Jim's recommendations of Kumar. Since they both recommended +I, this gives their recommendations semantic distances of 0 each. Figure 8.12 illustrates this example with Table 8.7.1 showing the resulting experience values.

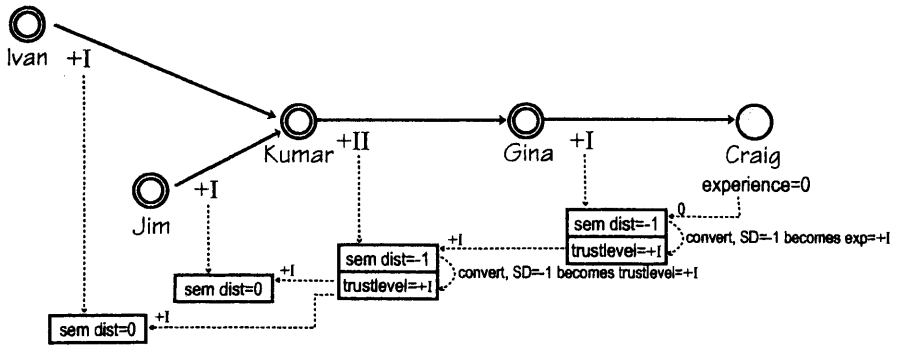


Figure 8.12: Experiences for all recommenders are derived from the original experience with direct prospect, backpropagated by recursive conversion of semantic distance to trust levels. "Sem dist" is the resulting semantic distance that forms the experience for the recommender that made the recommendation.

Recommender	Prospect	Recommendation	Experience with Prospect	Semantic Dist.
Gina	Craig	+I	O	-1
Kumar	Gina	+II	+I	-1
Jim	Kumar	+I	+I	0
Ivan	Kumar	+I	+I	0

Table 8.14: Example experiences for a chain after an interaction.

The schema *calcRecExperiences* formalises how experiences for recommendations are calculated. The input is a set of recommendations, the name of the direct trustee and the experience had with him. The result is a set of quadruples, each quadruple representing the recommender, context of the

experience, the trust level recommended by the recommender and the semantic distance between the experience with the actual trustee and this recommendation, i.e. the experience with the recommender.

calcRecExperiences assumes that all relevant direct recommendations and chains can be represented by a set of trees, with the root of each tree represented by the different direct recommendations that may exist in the truster's database of recommendations.

If the database of recommendations contains the recommendations as in Table 8.15, then the resulting tree will look like those in Figure 8.13. *calcRecExperience* uses this data structure to do a depth first search with tail recursion, calculating the experiences for the recommenders at each level before propagating each experience value lower down the branch to calculate the experience for that recommender's recommenders.

Recommender	Prospect	Context
a1	ad	d.c
a2	a1	r1.c
a3	a1	r1.c
a4	d	d.c
a5	a4	r1.c
a6	a5	r2.c
a7	a6	r3.c
a8	a6	r3.c
a9	a6	r3.c
a10	a4	r1.c
a11	a10	r2.c
a12	a10	r2.c

Table 8.15: Example recommendations.

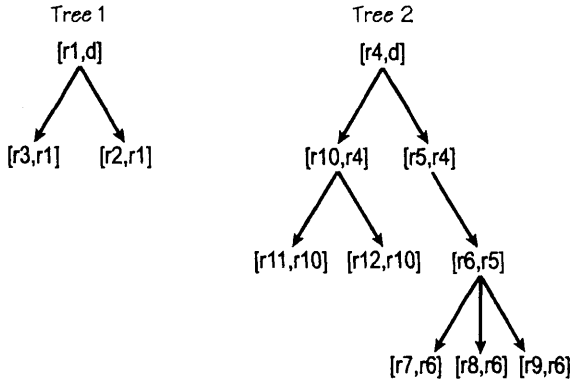


Figure 8.13: Tree data structure representing all direct recommendations and recommendation chains for prospect *d*. Nodes are recommendations where [r,p] represents recommendation from r about prospect p. Direct recommendation about prospect is at the root. Children of each node are recommendations about the recommender of that node.

sd2tv converts a given absolute semantic distance to its *TRUSTVALUE* equivalent, as in Table 8.12.

sd2tv

$$absoluteSD? : 0..\#TRUSTVALUE$$
$$result! : TRUSTVALUE$$
$$ord(result!) = \#TRUSTVALUE - absoluteSD?$$

Experiences with recommenders are stored in the *RExperienceSet* set containing members of type *RExperienceRec* (see page 168). The *addRExperience* function updates *RExperienceSet* and *getRExperiences* returns the set of recommender experiences for a given recommender and context.

addRExperience

$$\Delta RExperienceSet$$
$$newRec? : RExperienceRec$$
$$DExperienceSet' = DExperienceSet \bigcup \{newRec?\}$$

getRExperiences

$$\Xi RExperienceSet$$
$$forRecommender? : AGENTID$$
$$forContext? : Context$$
$$result! : \mathbb{P}\, SEMDIST$$
$$result! = \{rer : RExperienceRec \mid rer \in RExperienceSet \land (rer.agent = forRecommender?) \land (rer.context = forContext?) \bullet rer.semDist\}$$

calcRecExperience

$Recs? : \mathbb{P}\, Recommendations$

$trustee? : AGENTID$

$context? : Context$

$experience? : TRUSTVALUE$

$result! : \mathbb{P}(AGENTID \times Context \times TRUSTVALUE \times SEMDIST)$

$RecsForTrustee = \{ rec : Recommendation \mid rec.prospect = trustee? \land$
$\quad rec.context.contextID = context?.contextID \land$
$\quad rec.context.depth = context?.depth + 1 \}$

$RecsForTrustee = \emptyset \Rightarrow result! = \emptyset$

$RecsForTrustee \neq \emptyset \Rightarrow$
$\quad RecExps = \{ a : AGENTID;\ c : Context;\ tl : TRUSTVALUE;$
$\quad\quad sd : SEMDIST;\ rec : Recommendation \mid$
$\quad\quad rec \in RecsForTrustee \land rec.recommender = a \land$
$\quad\quad rec.trustLevel = tl \land sd = ord(experience?) - ord(rec.trustLevel)$
$\quad \bullet (a, c, tl, sd) \}$
$\quad RecRecExps = \{ a2 : AGENTID;\ c2 : Context;\ tl2 : TRUSTVALUE;\ sd2 : SEMDIST \mid$
$\quad\quad \exists a : AGENTID;\ c : Context;\ tl : TRUSTVALUE;\ sd : SEMDIST \bullet$
$\quad\quad\quad (a, c, tl, sd) \in RecExps \land$
$\quad\quad\quad (a2, c2, tl2, sd2) \in calcRecExperience(Recs?, a, c, sd2tv(\mid sd \mid)) \}$
$\quad result! = RecExps \bigcup RecRecExps$

8.7.2 Phase Transitions

Just as relationships in the direct category, after the end of each interaction the relationships with each recommender whose recommendations are used in that interaction are checked for possible phase transitions. The difference in the latter's case is in the conditions that govern phase transitions; recommendation context relationships base phase transitions on similarity measures or consistency or both.

For recommenders in the unfamiliar phase, i.e. first time recommenders, the absolute semantic distance between the first experience and recommended trust level decides which phase the relationship changes into. This approach is similar to how one might place value of first impressions. If we were recommended a builder for fitting our new kitchen by a friend, for the first time, and the job turned out to be horrible, then it is difficult to imagine ourselves asking that friend for the advice again.

For relationships in the Fragile phase, thresholds for number of previous experiences (in terms of absolute semantic distances) govern the changeover, in much the same way as the threshold mechanism for direct context relationships as shown on page 157. Thus, threshold policies such as the following can be asserted:

For any Fragile relationship in the r1.cServicePayment context, change the relationship to untrusted if the trustee produced either one experience with absolute semantic distance 4, or one at 3, or two at 2.

For illustration, consider our running example with Bob, whose unfamiliar and Fragile phase transition policies are shown in Table 8.16.

From Phase	To Phase	Thresholds				
		0	1	2	3	4
Unfamiliar	Untrusted	0	0	1	1	1
Unfamiliar	Fragile	1	1	0	0	0
Fragile	Untrusted	0	0	5	3	2
Fragile	Stable	3	5	0	0	0

Table 8.16: Example Unfamiliar and Fragile Phase Transition Policy for recommend context *r1.cServicePayment*. Thresholds are in absolute semantic distances. Stable phase transitions are shown in Table 8.17.

Assume that one of Bob's recommenders in the Fragile phase, Ulrich, has an experience set of the following:

$$\{1, -1, -1, 1, 1, 0, 2, 2\}$$

Bob removes the signs of the semantic distance values in the set to obtain:

$$\{1, 1, 1, 1, 1, 0, 2, 2\}$$

... which gives a tally of 5 for the absolute semantic distance value 1, 1 for the value 0 and 2 for the value 2. Thus according to Table 8.16 above, Ulrich's phase can be advanced to the Stable phase because his experience set meets the phase transition condition in the second Threshold column.

Policies for phase transition in unfamiliar and Fragile relationships are stored in the *UFRPTPolicySet* whose members are of type *UFRPTPolicyRec*. The Z specification to check phase transitions for recommenders in the unfamiliar or Fragile phase is given in *checkUFRPhase*. The functions *addUFRPTPolicy* and *getUFRPTPolicy* can be used to assert and retrieve the phase transition policies.

$$
\begin{array}{|l}
\textit{UFRPTPolicyRec} \\
\hline
\textit{context} : \textit{Context} \\
\textit{fromPhase}, \textit{toPhase} : \textit{PHASE} \\
\textit{ThresholdSet} : \mathbb{P}\,\mathbb{N} \mapsto \mathbb{N} \\
\hline
\forall\, tv_1, tv_2 : \textit{TRUSTVALUE};\ m, n : \mathbb{N} \bullet \\
\qquad ((tv_1, m) \in \textit{ThresholdSet}) \land ((tv_2, n) \in \textit{ThresholdSet}) \Rightarrow tv_1 \neq tv_2 \\
\hline
\end{array}
$$

$$\textit{UFRPTPolicySet} == \mathbb{P}\,\textit{UFRPTPolicyRec}$$

checkUFRPhase

$$
\begin{array}{l}
forRecommender? : AGENTID \\
forContext? : Context \\
trusteePhase : PHASE \\
EffectivePolicySet : \mathbb{P}\, UFRPTPolicyRec \\
\hline
trusteePhase = getPhase(forRecommender?, forContext?) \\
EffectivePolicySet = getUFRPTPolicy(forContext?, trusteePhase) \\
\\
\exists\, ep : UFRPTPolicyRec \bullet \\
\quad (ep \in EffectivePolicySet \\
\quad \land \exists\, threshold : \mathbb{N} \mapsto \mathbb{N} \mid threshold \in ep.ThresholdSet \bullet \\
\qquad \#\{ \forall x : \mathbb{N} \mapsto \mathbb{N} \mid x \in ((\mathrm{id}\, getRExperiences(forTrustee?, forContext?)) \\
\qquad\quad \lhd first(threshold)) \} = second(threshold)) \\
\Rightarrow setPhase(\mu\, PhaseRec \mid \\
\quad agent = forRecommender? \land context = forContext? \land \\
\quad currentPhase = ep.toPhase \land since = time())
\end{array}
$$

addUFRPTPolicy

$$
\begin{array}{l}
\Delta UFRPTPolicySet \\
newRec? : UFRPTPolicyRec \\
\hline
UFRPTPolicySet' = UFRPTPolicySet \bigcup \{newRec?\}
\end{array}
$$

getUFRPTPolicy

$$
\begin{array}{l}
\Xi UFRPTPolicySet \\
forContext? : Context \\
forPhase? : PHASE \\
result! : \mathbb{P}\, UFRPTPolicyRec \\
\hline
result! = \{ rptrec : UFRPTPolicyRec \mid (rptrec \in UFRPTPolicySet) \\
\qquad \land (rptrec.context = forContext?) \land (rptrec.fromPhase = forPhase?) \}
\end{array}
$$

If Bob had received a recommendation from an unknown (unfamiliar) recommender, say from an agent named Paul, then Paul's recommendation would have been ignored while calculating the reputation value for Craig. However that recommendation would have been retained by Bob so that he can evaluate the recommendation after interaction with Craig and decide whether he would like to initiate a relationship with Paul as recommender. If the recommendation from Paul had been 0 for Craig, for example, then this gives a semantic distance of 0. According to Bob's policy in Table 8.16, this allows Paul's relationship (currently in the unfamiliar phase) to advance to Fragile, thus

adding a new recommender that Bob can call upon for future recommendations.

For Stable relationships, consistency trust is used to determine phase transition. The use of the consistency threshold is driven by the requirement that once we have started to have trust in a recommender, we would like to know if he will continue to be reliable in the relationship. For this, phase transition policies such as the following can be asserted:

> *For any Stable relationship in the r1.cServicePayment context, change the relationship to untrusted if the trustee's current consistency trust is below 0.*

Table 8.17 shows Bob's policy for Stable recommend context relationships. Assume that Bob's previous experiences with Saul (who is a Stable phase recommender) is given in Figure 8.14.

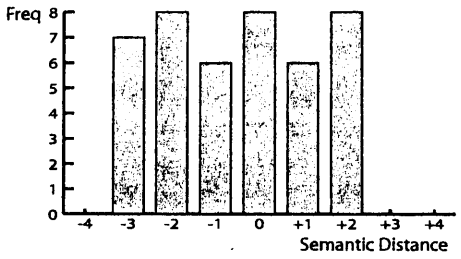


Figure 8.14: Example experiences for a Stable phase relationship.

Although the median semantic distance from Saul's experiences is 0, which is good, his consistency trust value is -1. Thus according to Bob's policy in Table 8.17, Saul's phase must be relegated to the untrusted phase.

From Stable to	Consistency
Untrusted	[-II,-I]
Fragile	0

Table 8.17: Example Stable Phase Transition Policy for recommend context *r1.cServicePayment*.

Policies for phase transition in Stable recommender relationships are stored in the *SRPTPolicySet* whose members are of type *SRPTPolicyRec*. The Z specification to check phase transitions for recommenders in the Stable phase is given in *checkSRPhase*. The functions *addSRPTPolicy* and *getSRPTPolicy* can be used to assert and retrieve the phase transition policies.

$$
\begin{array}{|l}
\textit{SRPTPolicyRec} \\
\hline
\textit{forContext} : \textit{Context} \\
\textit{toPhase} : \textit{PHASE} \\
\textit{threshold} : \textit{TRUSTVALUE} \\
\hline
\end{array}
$$

$$\textit{SRPTPolicySet} == \mathbb{P}\, \textit{SRPTPolicyRec}$$

addSRPTPolicy

$\Delta SRPTPolicySet$
$newRec? : SRPTPolicyRec$

$SRPTPolicySet' = SRPTPolicySet \bigcup \{newRec?\}$

getSRPTPolicy

$\Xi SRPTPolicySet$
$forContext? : Context$
$result! : \mathbb{P}\, PHASE \mapsto TRUSTVALUE$

$result! = \{ rptrec : SRPTPolicyRec \mid (rptrec \in SRPTPolicySet)$
$\quad \wedge (rptrec.context = forContext?) \}$

checkSRPhase

$forRecommender? : AGENTID$
$forContext? : Context$

$\exists ep : SRPTPolicyRec \bullet (ep \in getSRPTPolicy(forContext?)$
$\quad \wedge (getConsistency(forRecommender?) \leqslant ep.threshold)$
$\Rightarrow setPhase(\mu\, PhaseRec \mid$
$\quad agent = forRecommender? \wedge context = forContext? \wedge$
$\quad currentPhase = ep.toPhase \wedge since = time())$

We can now define a general purpose phase checking function for recommenders in arbitrary phases, in *checkRPhase*.

checkRPhase

$forRecommender? : AGENTID$
$forContext? : Context$

$getPhase(forRecommender?, forContext?) = phStable$
$\quad \Rightarrow checkSRPhase(forRecommender?, forContext?)$

$getPhase(forRecommender?, forContext?) \in \{phUnfamiliar, phFragile\}$
$\quad \Rightarrow checkUFRPhase(forRecommender?, forContext?)$

Below is the schema that is the actual entry point for post-interaction recording of experiences and checking of phases. It checks to see whether recommendations existed for the interaction in question

and if so calls the necessary state updating functions, *addRExperience* and *checkRPhase*.

updateState

$forTrustee? : AGENTID$
$forContext? : Context$
$experience? : TRUSTVALUE$
$recExperiences : \mathbb{P}(AGENTID \times Context \times SEMDIST)$

$addDExperience(\mu DExperienceRec \mid$
$\quad timestamp = time() \land context = forContext? \land$
$\quad agent = forTrustee? \land experienceLevel = experience?)$
$checkDPhase(forTrustee?, forContext?)$

$\exists rec : Recommendation \bullet$
$\quad rec \in Recommendations \land rec.prospect = forTrustee? \land$
$\quad rec.context.contextID = forContext?.contextID \land$
$\quad rec.contect.depth = forContext?.depth + 1 \Rightarrow$
$\quad\quad recExperiences = calRecExperiences(Recommendations, forContext?,$
$\quad\quad\quad forProspect?, experience?)$
$\quad\quad \forall a : AGENTID;\ c : Context;\ tl : TRUSTVALUE;\ sd : SEMDIST \mid$
$\quad\quad\quad (a, c, tl, sd) \in recExperiences \bullet$
$\quad\quad\quad addRExperience(\mu RExperienceRec \mid$
$\quad\quad\quad\quad timestamp = time() \land recommender = a \land$
$\quad\quad\quad\quad context = c \land trustLevel = tl \land semDist = sd)$
$\quad\quad\quad checkRPhase(a, c)$

8.8 Chapter Summary

In this chapter we looked at how Ntropi evaluates and uses individual recommendations, how recommendations are combined to form reputation for prospects and how experiences from direct interactions are turned into experiences with recommenders. We also looked at how Ntropi handles recommendation chains and the policies that govern its use.

An agent can recommend another agent with respect to a specific context or for recommending other agents for a specific context. The difference is determined by inspecting the *depth index prefix* of the context relevant to the recommendation. The context is not transitive, therefore trust in the same agent to recommend chains of depth n and $n + 1$ must be defined in separate relationships.

Since trust is subjective, the semantics of a trust level as defined by one agent may differ from the semantics of that defined by another agent. This difference is termed the *semantic distance* in Ntropi. Semantic distance is key in understanding how recommendations are used in Ntropi.

The semantic distance is obtained by taking the difference between a recommender's recommended trust level for a prospect and the actual experience had with the prospect. It is then stored as an

instance of experience with the recommender. By aggregating past semantic distances, Ntropi learns to match local semantics with that from the recommender. This semantic mapping between local and remote rating standards is used to 'translate' future recommendations into local semantics.

Ntropi also allows a simple measure of reliability of a recommender's recommendations. The reliability of a recommendation is based on how spread out the frequency of a recommender's past semantic distances is. The more spread out the frequencies, the less certain the semantics of his recommendation, and therefore, the less reliable his recommendations. A recommender's trustworthiness is based on this reliability measure, also called his *consistency*.

Recommendation chains in Ntropi are evaluated according to specific chain criteria, defined for each context. Each chain has a head, middle and last part. The head may contain more than one recommender, but they must all be known by the truster. The middle is a sub-chain of recommendations containing only unfamiliar recommenders. The last part is the direct recommendation about a prospect. Chain criteria can be specified for the head and middle parts, and must be met for the chain to be usable. A chain criteria specifying the maximum allowable chain length is also available.

Direct recommendations and recommendations is combined to form the *reputation* of the prospect. The combination algorithm uses a weighted sum approach, where the weights are based upon the recommenders' consistency.

After an interaction, Ntropi accepts experience values for updating the state of known agents in its database. As we mentioned above, the semantic distance defines an experience with a recommender.

The relationship phase of recommenders are also checked during this state updating stage. For unfamiliar and Fragile relationships, the rules for phase transition is defined in the same way as for relationships with direct contexts, as in the previous chapter: phase transition policy is defined in terms of thresholds on number of various semantic distances.

For Stable relationships, the phase transition policy is defined in terms of the recommenders' consistency. The truster defines consistency thresholds in the phase transition policy, and when a recommender's consistency falls below a threshold, the appropriate phase transition then takes place.

Chapter 9

Recommendation Protocol

Despite its huge size, our results indicate that the web is a highly connected graph with an average diameter of only 19 links.

Albert, Jeong and Barabàsi [AJB99]

A recommendation, as we have seen in previous chapters, is a message containing one agent's opinion about another. In this chapter we describe a simple *recommendation protocol* that agents can use to exchange recommendations with. We begin with the structure of protocol messages.

9.1 Message Structure

There are two message types in the recommendation protocol: *recommendation request* (RRQ) and *recommendation* (REC). They have the following structures.

RRQ

msgType : '*RRQ*'
requester : *AGENTID*
reqAddr : *ADDR*
context : *Context*
prospect : *AGENTID*
ttl : $\mathbb{N}$
senderPKID : *KEYID*
timestamp : *DATETIME*
expiry : *DATETIME*

The following are descriptions for each RRQ field.

msgType: Contains the string 'RRQ'.

requester: Name of the agent requesting the recommendation.

reqAddr: Address of agent requesting the recommendation.

context: The context of the recommendation requested, including the context ID and depth.

prospect: The name of the prospect about whom a recommendation is sought.

ttl: How far down the chain should the request be forwarded.

senderPKID: Identifier for the requester's public key for encryption of recommendation.

timestamp: When the request was sent.

expiry: The date beyond which the request can be ignored.

REC

msgType : *'REC'*
recommender : *AGENTID*
recAddr : *ADDR*
context : *Context*
prospect : *AGENTID*
trustLevel : *TRUSTVALUE*
senderPKID : *KEYID*
timestamp : *DATETIME*
expiry : *DATETIME*

The following are descriptions for each REC field.

msgType: Contains the string 'REC'.

recommender: Name of the agent who made the recommendation.

recAddr: Recommender's address for future contact from requester.

context: Context of the recommendation.

prospect: Prospect whom the recommendation is about.

trustLevel: The recommended trust level.

senderPKID: Identifier for the recommender's key used to sign the recommendation.

timestamp: When the recommendation was sent.

expiry: Date beyond which the recommendation should be discarded.

9.2 Recommendation Request

A request for a recommendation can be made by any agent. The agent may like to do this before transacting with an unknown agent, for example, and would like to know what his trusted recommenders think about the prospect, or what his reputation is. To request a recommendation, the requesting agent, or *requester*, creates an RRQ and sends it to his selected known recommenders.

For example, if Bob wants to send out a request about a prospect Dave, with respect to the context *cServicePayment*, then he would create the RRQ as in Table 9.2. He then selects the recommenders to send this RRQ to.

Field	Value
msgType	RRQ
requester	Bob
reqAddr	bob@p2pgrid.net
context	d.cServicePayment
prospect	dave@prospects.org
ttl	3
senderPKID	0xBEFD495D
timestamp	20041201
expiry	20041202

Table 9.1: Example recommendation request.

Recommenders are selected according to two parameters:

ContextID Recommender is in an existing relationship with the requester with respect to the context ID in the RRQ, regardless of context depth. This models the social trust principle where one trusts different sources for different contexts.

npMinPhase The minimum phase of relationship that a recommender must have with the truster, for the ContextID above. This is defined as an Ntropi policy. This is a simple approach to modeling the social trust principle where trust thresholds exist and one may choose to trust only when the relationship is above a certain trustworthiness threshold.

The *getRRQRecipients* schema contains the algorithm for selecting the candidate recommenders.

getRRQRecipients

$$\Xi PhaseSet$$
$$forContext? : CONTEXTID$$
$$minPhase : PHASE$$
$$prec : PhaseRec$$

$$minPhase = getPolicy(npMinPhase, forContext?)$$
$$result! = \{ r : AGENTID \mid prec.agent = r \land prec \in PhaseSet \land ord(getPhase(r, forContext?)) \geq ord(minPhase) \}$$

After sending out the RRQs to his selected recommenders, Bob then waits for the recommendations and then carries out any required recommendation processing. The timeout period will depend on the urgency of the recommendations. Received recommendations are stored in the *RecommendationSet* database, replacing any old recommendations, if any, that contain the same recommender, context and prospect. Recommendations are also checked first for its expiry date and expired recommendations are simply discarded.

9.3 Forwarding Recommendations

Upon receiving an RRQ, an agent may choose to act on it in one of two ways: forwarding or replying. An agent who received an RRQ may choose to forward the RRQ to his trusted recommenders if he is not able to reply with a recommendation. Before forwarding an RRQ, the agent first checks the *ttl* value in the RRQ to ensure that it has not reached its limit. If it is zero then it has reached its limit and the RRQ will not be forwarded. If it is greater than zero then the agent forwards the RRQ according to the following steps:

1. Decrement the *ttl* value in the RRQ by 1.
2. Select the recommenders to forward the RRQ to by calling *getRRQRecipients*.
3. Forward RRQ to selected recipients.

9.4 Replying with Recommendations

If the agent receiving an RRQ is able to reply to the RRQ then he creates and sends a Recommendation back to the requester. For example, if the RRQ from Bob reached Mark, who does have a recommendation about Dave for the requested context, then he creates the REC message as in Table 9.4 and sends it to Bob.

Field	Value
msgType	REC
recommender	Mark
recAddr	mark@recservices.com
context	d.cServicePayment
prospect	dave@prospects.org
trustLevel	+I
senderPKID	0xAFFE341B
timestamp	20041201
expiry:	20050601

Table 9.2: Example recommendation.

9.5 Privacy

There may be instances when the requester does not want to reveal that he is requesting recommendations about a prospect, perhaps for reasons of privacy. For example, within a small community where mutual trust is expected, enquiries into a community member's trustworthiness may be interpreted

as distrust, which may taint the bond that binds the community together. The social significance of this was discussed in §2.7.2.

A limited form of anonymous RRQ can be achieved by simply leaving blank the *requester* and *reqAddr* fields of the RRQ. This way, when the RRQ is forwarded, any recommendation can only be returned by replying back to the last forwarder of the RRQ. It is limited because: 1) the first recommender in the chain after the requester knows the requester, and 2) there is a path between the requester and direct recommender through all intermediate forwarders. Thus, the anonymity can be broken if the first agent the RRQ was sent to reveals the requester's address, or when there is collusion among all the recommenders in the chain.

A trust context for evaluating a recommender's trustworthiness with respect to respecting anonymity protocols may be required for requesters who must transmit anonymous RRQs so that he can select only those recommenders that he trusts not to reveal his identity.

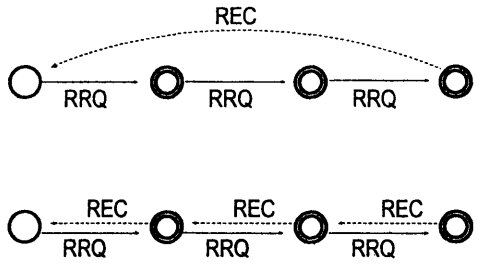


Figure 9.1: If an address is provided in the RRQ, the recommender can return a recommendation directly to the requester (top). Without an address, the recommendation must be returned along the recommendation chain (bottom).

The recommender may also want to keep his communication with the requester private, as it may not be desirable to make public one's opinion about another party. This can be done by encrypting the recommendation with the requester's public key referenced with the *senderPKID* in the RRQ. However, the recommender may want to assess the requester's trustworthiness with respect to non-disclosure before sending out the recommendation if secrecy is an important criteria.

Note that a public key can be used as an identifier to discover the identity of the requester. This may not be a problem when the exchange is between two mutually known and trusting parties, but may be an issue when the requester and recommender are strangers.

9.6 Chapter Summary

In this chapter we have outlined a simple recommendation protocol that can be used to request and transmit recommendations. We also discussed privacy considerations in this protocol and how limited anonymity can be achieved by the requester. This anonymity relies of the trustworthiness of intermediate recommenders with respect to non-disclosure of the requester's address. 'Eyes only' recommendations can also be created by encrypting recommendations with the requester's public key. However, this may be at the cost of the requester's anonymity, as his public key may be used to discover his identity.

Chapter 10

Bootstrapping and Database Maintenance

There are two lasting bequests we can give our children: One is roots, the other is wings.
Hodding Carter

Although this model allows agents to learn about new recommenders and other agents completely from scratch, i.e. without any prior experience or trusted recommenders, this is not recommended. When a new agent is created, the agent is faced with a high degree of uncertainty about other agents it may encounter. This is because it will be unable to distinguish between trustworthy and untrustworthy agents. This makes the agent vulnerable to manipulation, as is any complete newcomer to any community. There is always the chance that a rogue agent may take advantage of the unwitting newcomer by pretending to offer 'assistance for malicious hidden motives. In the real world, there are resources to guide newcomers into any field. For example, travel guides give recommendations to travelers on aspects of a particular destination. To reduce uncertainty and risk, it is recommended that new agents are equipped with a number of positive trust relationships in its database so that initial interactions can be made with already trusted parties or those recommended by already trusted recommenders.

This issue is more important for artificial agents than human agents because new artificial agents are inherently less 'experienced than human agents. An artificial agent, however, has already at least one default recommender: its human owner. Thus, there is no excuse for agents to be released into virtual communities with complete uncertainty, unless, of course, the intention is to seek out untested avenues. For such exploratory goals, it is recommended that deployed agents are robust and resilient to malicious encounters and risky environments.

With each experience and receipt of recommendation, data will accumulate in the databases. Some maintenance may have to be carried out on these databases, purging old and expired data. This is for two reasons: 1) memory management, and 2) to ensure stale information is not used in trustworthiness evaluations.

The limit placed on the amount of memory used will depend on the application and available resources. There may be applications where a high number of interactions in a short period of time may occur, such as in network routers or stock trading systems. In others, less frequent interactions may be the norm, such as buying flowers from an online florist. In addition, the distribution of inter-

action frequency over time may be iregular in nature, such as peak periods of web traffic during the day.

Memory management, however, also includes analysing tradeoffs between conserving memory space and having enough information to make trust evaluations with. History is essential to reputation systems and to the stability of markets that use it [JHF03]. However, research has shown that too much history may have a negative effect on incentivising positive behaviour. For example, Holmstrom showed that the professional who had accumulated enough good history and also who is being rewarded based on his entire track record, may lose incentive to work hard to maintain his track record (i.e. there is the tendency to "rest on one's laurels") [Hol99]. If, however, old observations of behaviour is discounted, incentives to work towards maintaining a positive track record that is current and recent remain constant. In Ntropi, we exploit freshness of information so that older experience data and recommendations can be purged from the databases.

Below we provide an audit of databases and policy variables within the framework and suggest default bootstrapping data for them. The next section discusses specific areas of the model where regular maintenance should be carried out.

10.1 Bootstrapping

There are several databases in use in Ntropi. These are used to store policy parameters, agent specific information and recommendations. Below we look at each and describe example default bootstrapping database entries.

10.1.1 Ntropi Policy Parameters

Blind Trust Blind Trust overrides any trustworthiness evaluation and always uses that specified trustworthiness value (page 150). Default blind trust entries may be based on attributes or agent name, and can be in conjunction with a given context. The following specifies that all agents from the domain 'cs.ucl.ac.uk' should have a trustworthiness value of +II for context *r1.cCSTextbookAuthor*.

$$addPolicy(npBlindTrust, r1.cCSTextbookAuthor, (attrib, `ucl.ac.uk', +II))$$

The following specifies that **@dailyplanet.com* should have a trustworthiness value of -II for context *d.cWorldNewsReport*.

$$addPolicy(npBlindTrust, d.cWorldNewsReport, (name, `* \bullet thesun.com', -II))$$

General Trust Disposition This models an agent's risk aversion, and is used when faced with a choice trustworthiness value to choose from (see page 145). Value is one of *gtdMedian*, *gtdRiskAverse* or *gtdRiskSelective*. The following specifies that the agent should have a risk averse disposition when evaluating prospects for the context *cInvestmentAdvice*.

$$addPolicy(npGenTrustDisp, cInvestmentAdvice, gtdRiskAverse)$$

General Classifiers Classifiers are strategies for generalising agent-neutral past experiences (page 143). The following specifies that the Context Experience classifier be used, and if that fails to return a result, then the Stereotype classifier.

$$addPolicy(npClassifier, d.cServicePayment, (clContextExperience, clStereotype))$$

Direct Phase Transition Thresholds This governs when a trust relationship should change from one phase to another (page 143). It is based on a threshold of a number of experiences, of a certain level, had since the last phase transition. The following states that all Fragile direct relationships in the *d.cServicePayment* context change to the stable phase if the agent has achieved at least 20 experiences of level 0, or 10 experiences of level +I or 5 experiences of level +II.

$$addPTPolicy(\mu\, PTPolicyRec \mid context = d.cServicePayment \wedge fromPhase = phFragile \wedge toPhase = phStable \wedge ThresholdSet = \{(-II, 0), (-I, 0), (0, 20), (+I, 10), (+II, 5)\})$$

Stable Phase Experience Weights In stable relationships, negative experiences have less of an impact than in Fragile relationships (§2.7.2). In Ntropi we model this by applying different weights to different experience levels (page 160). The following defines weightings as shown in Table 10.1 for context *cServicePayment*.

$$setWeighting(cServicePayment, \{(-II, 1), (-I, 1), (0, 3), (+I, 4), (+II, 6)\})$$

Experience level	$-II$	$-I$	0	$+I$	$+II$
Impact weighting	1	1	3	4	6

Table 10.1: Example stable phase experience weights.

Phase Equivalence This policy relates to chain evaluation, when a minimum number of stable chain heads must be present (page 174). Fragile phased relationships can be used if there are insufficient stable ones to meet the criteria. The phase equivalence policy is a ratio which states how many Fragile relationships should be used in place of stable ones, for a given context. The following states that for the context *cX509Issuer*, fragile to stable ratio for chain heads is 3:1.

$$addPolicy(npPhaseEquiv, cX509Issuer, (3, 1))$$

Chain Criteria Chain criteria specifies minimum trustworthiness values that various entities in a chain must meet for the chain to be valid (page 175). The following specifies a chain criteria as in Table 10.2.

$$addChainCriteria(cFileQuality, (\mu\, ChainCriteriaRec \mid headConsTrust = +I \wedge headCount = 2 \wedge intReputation = +I \wedge maxRecDepth = 3))$$

Recommender Consistency Weights When combining recommendations, Ntropi takes into account the trustworthiness of each contributing recommender by applying user-defined weights,

Chain criteria for context: *cFileQuality*	
Criteria	Parameter
Head	*headConsTrust* $\geqslant$ +I
	headCount $\geqslant 2$
Intermediary	*intReputation* $\geqslant$ +I
Max Chain Depth	3

Table 10.2: Example chain criteria.

called *conweights* (page 181). Weights are assigned to each trustworthiness level, for each context. The following sets conweights as in Table 10.3 for context *cServicePayment*.

$$addConweight(cServicePayment, \{(-II, 0), (-I, 0), (0, 1), (+I, 2), (+II, 5)\})$$

Consistency Trust	-II	-I	0	+I	+II
Weight	0	0	1	2	5

Table 10.3: Example conweights for context *cServicePayment*.

Recommender Phase Transition Thresholds As the Direct Phase Transition Thresholds above, this parameter governs when phase transitions occur. For recommenders, one of two parameters is used, depending on whether the relationship is stable or not. Transition from a stable phase is based on the recommender's consistency trustworthiness (page 196). The following states that the stable relationship should be changed to untrusted if consistency trust is below -I.

$$addSRPTPolicy(\mu\, SRPTPolicyRec \mid forContext = r1.cServicePayment \wedge toPhase = phUntrusted \wedge threshold = -I)$$

For Unfamiliar and Fragile relationships, phase transition is based on thresholds on the number of specific levels of experiences. The following states that all Fragile relationships under context *r1.cServicePayment* change phase to untrusted if agent has at least 3 experiences of level -II or 5 at level -I.

$$addUFRPTPolicy(\mu\, UFRPTPolicyRec \mid context = (\mu\, Context \mid contextID = cServicePayment \wedge depth = 1) \wedge fromPhase = phFragile \wedge toPhase = phUntrusted \wedge ThresholdSet = \{(-II, 3), (-I, 5), (0, 0), (-I, 0), (-II, 0)\})$$

Recommender Protocol Minimum Phase This policy is used in the recommendation protocol. It specifies the minimum phase of relationship with a recommender for that recommender to be sent a recommendation request (RRQ). The following instructs Ntropi to send RRQs for the context *cAudioQuality* only to stable phase recommenders.

$$addPolicy(npMinPhase, cAudioQuality, phStable)$$

10.1.2 Trustee Specific Information

Agent Relationship Phases *PhaseSet* contains all known trust relationships (page 142). Relationships can be added and made known to Ntropi by the following example, which adds a stable relationship with agent *Bob* in context *d.cServicePayment*.

$$setPhase(\mu\, PhaseRec \mid agent = Bob \wedge currentPhase = phStable \wedge since = 20041201 \\ context = (\mu\, Context \mid contextID = cServicePayment \wedge depth = 0))$$

Agent Attributes Agent attributes are used for stereotyping Unfamiliar agents (page 147). Known agent attributes can be added to the database, *AttributeSet*, as in the following example, recording the domain attribute emphucl.ac.uk for agent *Bob*.

$$addAttribute(\mu\, AttributeRec \mid agent = Bob \wedge attrib = \text{`}ucl.ac.uk\text{'})$$

Direct Experiences Direct experiences are recorded in the *DExperienceSet* database (page 146), and can be added as in the following example, which record an experience with agent *Bob* of level +I in context *d.cServiceAvailability*, dated 1 December, 2004.

$$addExperience(\mu\, DExperienceRec \mid timestamp = 20041201 \wedge agent = Bob \wedge \\ context = (\mu\, Context \mid contextID = cServiceAvailability \wedge depth = 0) \wedge \\ experienceLevel = +I)$$

Recommend Experiences Recommend experiences are recorded in the *RExperienceSet* database (page 168), and can be added as in the following example, which record an experience with agent *Bob* of semantic distance -3 for his recommendation of +I in context *r1.cEVendor*, dated 1 February, 2004.

$$addRExperience(\mu\, RExperienceRec \mid timestamp = 20040201 \wedge agent = Bob \wedge \\ context = (\mu\, Context \mid contextID = cEVendor \wedge depth = 1) \wedge \\ trustLevel = +I \wedge semDist = -3)$$

For each known relationship, there has to be an entry in the phases database, *PhaseSet*, and one or more entries in the direct or recommend experiences database, *DExperienceSet* or *RExperienceSet*, for direct or recommend relationships respectively. This is because if a relationship exists then Ntropi assumes prior experience with the agent. Conversely, if there are experiences with an agent then Ntropi assumes that there is an existing relationship with that agent.

Entries in the agent attributes set, *AttributeSet*, are optional.

10.1.3 Recommendations

Recommendations are stored in the *RecommendationSet* database (page 168). The following example stores a recommendation from *Alice* of +II about *Bob* for context *d.cServiceQuality*, dated 25 December, 2003.

$addRecommendation(\mu Recommendation \mid timestamp = 20031225 \land recommender = Alice \land$
$context = (\mu Context \mid contextID = cServiceQuality \land depth = 0)$
$prospect = Bob \land trustLevel = +II)$

10.2 Maintenance

Maintenance of databases involve purging expired database entries. This applies to three databases: 1) direct experiences (*DExperienceSet*), 2) recommend experiences (*RExperienceSet*) and 3) recommendations received (*RecommendationSet*). For specification purposes, we define the freshness conditions for these databases in Z, instead of algorithms for specifically searching for and purging records.

Direct experiences Maintenance of *DExperienceSet* is dictated by how much of the trustee's history the truster wants to include when evaluating the trustee's trustworthiness. This may vary from one context to the next and so this history cut-off threshold is specified per context. This threshold may be specified as an age (*coAge*), where experiences older than this age are purged, or as the maximum number of records to keep in the database at any given time (*coCount*). The choice will depend on the application and context – *coCount* may be preferable where there is a high volume of interactions within a short period of time. These history cut-off parameters are defined as Ntropi policies, and can be defined using the *addPolicy* function. The following states that for context *d.cGoodsDelivery*, direct experiences more than 730 days old are to be discarded.

$$addPolicy(npHistCutoff, d.cGoodsDelivery, (coAge, 730))$$

The following schema adds the freshness condition to the *DExperienceSet* definition.

$$DExperienceSet == \mathbb{P}\, DExperienceRec$$

$$\begin{aligned}
&\forall e : DExperienceRec \mid e \in DExperienceSet \bullet \\
&\quad first(getPolicy(npHistCutoff, e.context)) = coAge \Rightarrow \\
&\qquad e.timestamp \geq (today() - second(getPolicy(npHistCutoff, e.context))) \\
&\quad first(getPolicy(npHistCutoff, e.context)) = coCount \Rightarrow \\
&\qquad \exists CtxtSet : \mathbb{P}\, DExperienceRec \mid CtxtSet \subseteq DExperienceSet \bullet \\
&\qquad\quad (\forall x : DExperienceRec \mid x \in des2 \bullet x.context = e.context) \land \\
&\qquad\quad (\#CtxtSet \leq second(getPolicy(npHistCutoff, e.context)))
\end{aligned}$$

Recommend experiences The freshness condition for experiences with recommenders is exactly the same as for direct experiences, except that it is for the set *RExperienceSet*.

$$
\begin{array}{|l}
RExperienceSet == \mathbb{P}\, RExperienceRec \\
\hline
\forall e : RExperienceRec \mid e \in RExperienceSet \bullet \\
\quad first(getPolicy(npHistCutoff, e.context)) = coAge \Rightarrow \\
\qquad e.timestamp \geq (today() - second(getPolicy(npHistCutoff, e.context))) \\
\quad first(getPolicy(npHistCutoff, e.context)) = coCount \Rightarrow \\
\qquad \exists CtxtSet : \mathbb{P}\, RExperienceRec \mid CtxtSet \subseteq RExperienceSet \bullet \\
\qquad\quad (\forall x : RExperienceRec \mid x \in des2 \bullet x.context = e.context) \wedge \\
\qquad\quad (\#CtxtSet \leq second(getPolicy(npHistCutoff, e.context)))
\end{array}
$$

Recommendations An age cutoff policy is specified for recommendations received. Furthermore, recommendations also have expiry dates specified by the recommender which are embedded in the recommendation itself. Recommendations that are past this expiry date or those that are older than the cutoff age are purged from the database.

Recall that on page 168 we defined a schema for the recommendations database, *RecommendationSet*, which includes recommendation uniqueness conditions. We now extend this schema to include the freshness conditions above.

$$
\begin{array}{|l}
RecommendationSet == \mathbb{P}\, Recommendation \\
\hline
\forall r1, r2 : Recommendation \mid r1 \in RecommendationSet \\
\quad r2 \in RecommendationSet \wedge r1 \neq r2 \\
\quad \Rightarrow \neg\, (r1.recommender = r2.reocmmender \wedge r1.prospect = r2.prospect \\
\qquad \wedge\, r1.context = r2.context) \\
\forall r : Recommendation \mid r \in RecommendationSet \\
\quad (r.expiry > today()) \wedge (r.timestamp \geq (today() - getPolicy(npRecAge, r.context)))
\end{array}
$$

It is uncertain at the moment which different values for the freshness parameters are 'good'. Without further understanding of the application within which the model will be applied, we can only recommend general principles for their use, i.e. using these parameters to reduce memory versus having enough experience history to calculate trust values with. Further tests and simulations will be required to study the sensitivity of these parameters and their effect on trust calculation.

In general, there are arguments for both holding long-term and only short-term history. Since trust very much depends on previous knowledge, it is reasonable to assume that the longer the history the better our trust judgement will be about the prospect. However, this requires more memory use so may not be suitable for devices with resource constraints, such as mobile nodes in a pervasive environment. Furthermore, we would like to know what a prospect's recent behaviour is like, and therefore, we are mainly concerned about recent history. What is 'recent' is then an open question as different applications may have their own definition for this. These issues were discussed in the introductory paragraphs of this chapter.

Perhaps a balance between storing long-term and short-term history can be achieved by only carrying a summary of the long-term history but a more detailed recent history. This will be a subject of future work.

10.3 Management of Policy Parameters

As can be seen in the preceding pages, there are a large number of parameters that an Ntropi user may have to consider. There are in fact 14 parameters for Ntropi alone, and this does not include those that the application may add. Thus, management of these parameters may become complex in real systems.

However, the parameters presented in this work are just the *possible* dimensions for the straw man model that we have proposed. In actual implementations, these will have to be collapsed into a more manageable set. Some of the parameters may be delegated to the user to manage and others possibly hard-coded into software by the engineers who build it. Which to include or exclude is an open question and likely to be dependent on the application domain.

A further implementation challenge is the method of input and ensuring that the parameters are valid. Some of the parameters in Ntropi, such as the recommender consistency weights (conweights) must be defined by the user and a suitable user interface must be devised in order to reduce the complexity of managing these parameters. Furthermore, although we have selected a discrete trust level scale with only five values to reduce complexity for the user, the actual presentation for these levels can still be problematic. For example, which representation will be more comfortable for users, integers or Roman numerals? Collaboration with the Human-Computer Interface community will be useful here.

Additionally, care must be taken to ensure that the parameters from user input are complete, valid and consistent. An example where this can be an issue are the phase transition policies, such as the one shown in Table 8.16 on page 194, where a collision in transition conditions may occur, shown, for example, in Table 10.4 under column '2'. This problem, however, can be easily rectified by adding a policy validator in applications of the model to check for such human errors. The Ntropi policy parameters that will require consistency checks are:

- Direct Phase Transition thresholds
- Chain Criteria
- Recommender Consistency Weights
- Recommender Phase Transition threshold

From Phase	To Phase	Thresholds				
		0	1	2	3	4
Fragile	Untrusted	0	0	5	3	2
Fragile	Stable	3	5	4	0	0

Table 10.4: Example collistion in Phase Transition Policy.

10.4 Chapter Summary

When instantiating Ntropi in applications, its databases must be populated with entries to help bootstrap the trust evaluation process. This is necessary as Ntropi works on the principle of evaluating

the unknown from known data. Without bootstrapping information, an agent using the Ntropi model will be paralysed when asked to evaluate trustworthiness.

Due to resource constrains and the need to work with fresh information, Ntropi's databases must be maintained, purging expired and old data.

In this chapter we looked at how policy variables, agent specific information and recommendations can be bootstrapped. Policy variables were: Blind Trust, General Trust Disposition, General Classifiers, Direct Phase Transition Thresholds, Stable Phase Experience Weights, Phase Equivalence, Chain Criteria, Recommender Consistency Weights, Recommender Phase Transition Thresholds and Recommender Protocol Minimum Phase. Agent specific information are the databases that contain experiences, *DExperienceSet* and *RExperienceSet*, and recommendations, *RecommendationSet*.

For maintenance, freshness conditions are used. This is specified in the Ntropi policy. We then extended the Z schemas for the definition of the three databases that require maintenance, *DExperienceSet*, *RExperienceSet* and *RecommendationSet*, to include freshness conditions to them.

Chapter 11

Threat Analysis

"Why a revolver?" he asked.
"It helped me to trust in people," the Englishman answered.

Paulo Coelho

There is some amount of risk involved in our everyday activities. Such risks may involve misunderstandings, accidents and intentional malicious threats. Trust and reputational information help us to manage these risks. With the case of intentional malicious acts, however, the agents who carry them out can and will try to reduce the potency of these tools. For example, if a restaurant that uses stale or expired food ingredients can stop word getting round in the community about it's bad food then it can continue to do so without fear of being discovered. For a community to remain healthy (not just by not eating bad food, but also by discouraging bad behaviour) it's members must be aware of such threats so that the impact of these threats can be minimised.

Threats to trust models [Ope03] reduces the model's effectiveness as well as users' trust in the model itself. In this chapter we will be looking at how malicious agents may try to subvert a trust model, how a benign agent may react to such subversions and the dynamics of a community of agents in the light of such threats. Note that we are only concerned with motivations and strategies at the 'reputation layer', not the application layer. The latter may include primary motivations such as profit, which drives secondary motivations or goals such as attacking the reputation of another agent at the reputation layer.

These attacks, we believe, will be carried out by intelligent agents with the ability to evolve strategies according to the target environment and entities. The goal of this chapter is to inform about possible strategies available to attackers. This can then be developed in future work to model intelligent malicious agents in simulated attacks.

We will begin by assuming that each agent interacts within a community of agents. The role of our trust framework is to allow agents to identify and purge anti-social members as quickly as possible while retaining the community's benefits of membership. We now look at the concept of community in detail.

11.1 Community

A community consists of two or more communicating agents grouped together by a common interest. Members within a community are there to help each other towards a common goal, such as sharing quality information and recommendations about good hill-walking routes in the country. As such, members in a community must be reachable through paths of recommendations. If we represent community members as vertices and recommendations paths as edges in a graph, such a community may look like those in Figure 11.1 below.

Specifically, a community:

1. Is a *directed graph*, because trust relationships (which channel recommendations) are unidirectional.
2. Is *unilaterally connected* [LL92], i.e. given two nodes x and y in the graph, x must be reachable from y, but not necessarily the other way around.
3. May have cycles.

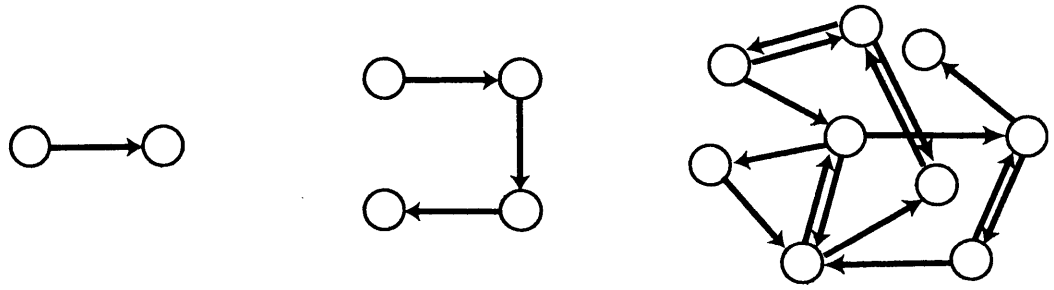


Figure 11.1: Examples of communities. Arrows indicate direction of trust relationship, from truster to trustee.

As mentioned above, what brings agents together within a community is a common interest in seeking reliable sources and recommendations about a particular situation, such as the hill-walking example given above. We will refer to this common area of interest as the *context* of the community. We will see later that this is also the same context that trust relationships are built around. It is also natural for individuals to participate in more than one community as most people will have more than one area of interest. This will also be true for agents in our model.

There will be no predefined hierarchy within the community and we will assume dynamic topology within each community – relationships form or die and members join and leave arbitrarily. We will assume also arbitrary life span for each community.

In addition, a community may be formed structurally or by emergence. A **structural** community has clear boundaries of membership. Example structural communities include ad-hoc networks and employees of University College London (UCL). Structural communities have the following properties:

- Membership of structural communities can be ascertained objectively, e.g. it can be determined whether a person is a UCL staff or not, and there is a distinct membership process.

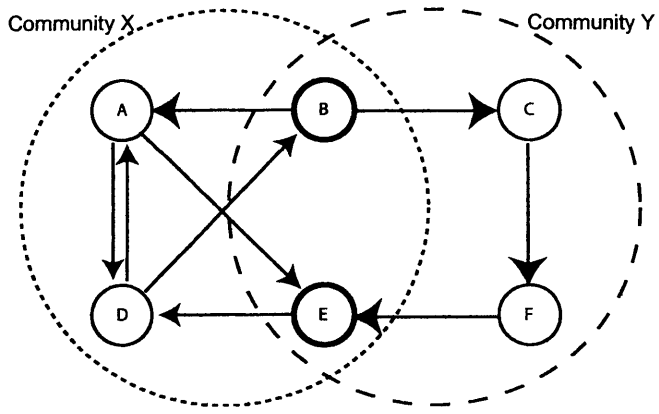


Figure 11.2: Social butterflies: Agents B and E are members of communities X and Y, communities formed from different contexts of trust relationships.

- Membership of the community is governed by either an administrator or standard protocol. The UCL staff registrar and a node announcing itself to the routers in an ad-hoc network are such examples.

An **emergent** community takes shape from the individual trust relationships between it's members. The relevant relationships that make up the community are those that lay within the same context. For example, a community for those sharing recommendations about classic cars will be connected through trust relationships within the "classic cars" context.

11.2 Attacks on a community

A community may come under several forms of attacks. Reasons that may motivate attacks on communities include:

- To make attacks on individual members of the community easier.
- To destroy the community or reduce it's influence, perhaps because that community favours the competition.
- To attack a specific community member indirectly by reducing the reputation of the community, thereby reducing the member's reputation via his affiliation with the community.

Threats to communities can be roughly grouped into three kinds – exploitation, weakening and destruction. We will look at these in turn below.

11.2.1 Exploitation

In exploitation threats, the motive of the malicious agent, whom we will refer to as Rogue from now on, is to gain without paying. In other words Rogue's intention is to cheat as many members of the community as possible. There are various approaches available to Rogue and these will be described below.

Positive Collusion with Benign Accomplice

Here a group of agents collude to make positive recommendations about each other to boost their reputations [MGM04, KSGM03]. Although a group can be made up of just two agents, larger groups will be more effective.

Prior to mounting an attack, members of the colluding group first generate good recommendations about each other's trustworthiness as recommenders, i.e. in the form of "Agent X is a good recommender". This will build up the group members' reputation as recommenders. Once the desired reputation level is attained the group members then begin generating positive recommendations about selected members from the group (e.g. "Agent Y is a trustworthy cash borrower"). These selected members are the ones who will be mounting an exploitative attack on benign agents once their reputations reach a level which enables them to gain trust from community members. The good reputation is then abused, for example purchasing items on credit and then reneging on repayment. The attackers continue abusing until they are expelled from the community.

Their accomplices remain in the community. After the attack, the attackers may then try to return to the community with a new pseudonym and repeat this process.

This attack can be prevented from recurring by lowering the reputation of the accomplices after an attack.

Negative Collusion with Benign Accomplice

The collusion mechanism for building up reputation among colluding group members as good recommenders is the same as above. This time, however, the goal is to spread negative recommendations about a benign agent (or a number of benign agents) outside the colluding group so that the victim gains a negative reputation in the community [MGM04, KSGM03].

Various strategies [KSGM03] can be employed here, described below:

Constant release In this strategy the malicious recommenders send out negative recommendations about the victim every time a recommendation is sought about him.

Random release Here the malicious recommenders will randomly break the negative recommendation consistency by occasionally giving out positive recommendations about the victim. The goal is to water down the negative recommendations to hide the attack, and to instill doubt in the recommendation requestor about the victim. This attack will take longer than the previous strategy but it reduces the chances of an attack being detected early on.

Selected release When the desired goal is to create tension between two agents then the malicious agents can choose to give out recommendations only to certain recommendation requestors, i.e. the ones that should distrust the victim.

Exploiting Reputation Time Decay

This threat may arise in systems where the certainty or validity of reputations decay with time (see eBay example in [Boy02]). Once Rogue gains enough reputation this reputation is abused by attack-

ing one or several benign agents within a short period of time. After the attack, Rogue returns to behaving well, although this time his reputation has been reduced because of the attack. With time, Rogue positive behaviour will outweigh his old negative reputation and he will then have another chance to build up a positive reputation to carry out another attack. The process is repeated.

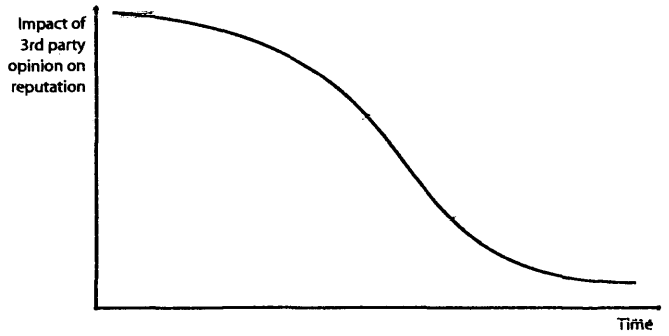


Figure 11.3: The impact of an opinion by a recommender decays lessens with time. A malicious agent can exploit this with a wait-attack-wait cycle.

However, too strong an attack and Rogue risk being completely distrusted by the community members, jeopardising his opportunities for future attacks. Therefore there is a possibility that Rogue will spend a period of time learning about the optimum level of attack in the community, one which will maximise his gains while still allowing him to remain in the community for future attacks.

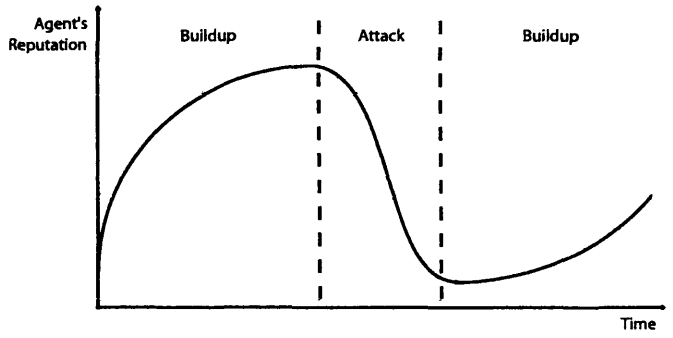


Figure 11.4: Rogue builds up reputation then mounts an attack. Rogue continues with good behaviour to build up reputation in preparation for another attack.

Masquerading

When processing reputational information multiple recommendations will tend to reinforce each other if they give similar recommendations. The collusion attacks above can be used for this purpose. However, if a malicious agent is acting on his own, he could form his own colluding group by having several pseudonyms, masquerading as different agents. This is also known in peer-to-peer networks as the Sybil Attack [Dou02].

11.2.2 Weakening

The degree of *connectedness* reflects the potential for reputational information to be pervasive throughout the network. A better connected community has more channels of communication, which means that there is more redundancy in communication channels giving reputational information a

better chance to spread [Put95]. In our model the strength of a community is based on the degree of connectedness, or "network density" in social theory parlance, of that community – the higher the degree, the stronger the community.

Weaker communities are more vulnerable to attacks because it is easier to break them apart. By splitting the community Rogue can then mount an attack on one of the broken away segments of the community without news of his malicious actions reaching the other in time (or ever). This gives Rogue time to attack the other segments too. It is thus in the interest of weaker communities to rapidly build as many trust relationships as possible to reduce the impact of an attack.

In Figure 11.2 for example, community Y can be split by just severing the link between C and F, while it will require more links to be broken to split community X. To sever a link can either mean to physically make the nodes unreachable, or to make one agent distrust another. As an example of the latter, Rogue may set up the attack by first building up a really good reputation within the community perhaps by repeatedly displaying outstandingly trustworthy behaviour. Once sufficient trust is reached he can then start bad-mouthing F by sending negative recommendations about him to C. If recommendations from Rogue is enough to convince C about F being untrustworthy then this may be enough to sever C's trust in him. As a result the community will be split because C will not trust F anymore, as shown in the diagram below. R can now exploit the two communities separately, now with a decreased probability that he will be discovered after the first exploitative behaviour.

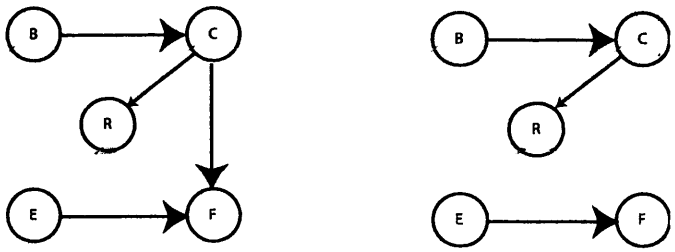


Figure 11.5: Weakening a community. Left: R builds C's trust in himself. Right: R abuses trust by bad-mouthing F to C and breaking up trust between C and F.

11.2.3 Destruction

The goal of an attacker in this situation is to render the target community untrustworthy as a source of good reputational information, and ideally to dissolve it altogether. There are two approaches that the attacker can take: insider and outsider attacks.

Insider Attack

In mounting an insider attack, Rogue starts by joining the target community (the community he is mounting an attack on). Once a member, Rogue then begins behaving abusively towards members in other communities (or the "deceived" communities). With time and repeated abusive behaviour, Rogue will acquire a negative reputation from himself. This is the intended outcome – if it is known that Rogue is a member of the target community then his bad reputation can potentially be transferred to other members of the target community. Effectively, Rogue would have transferred his bad reputation to the community through merely being affiliated to it.

The disadvantage of this attack is that Rogue must be able to sacrifice his identity as it is likely that the negative reputation earned is irreparable. This attack is also dependent on inter-community links. If the agents in a deceived community that Rogue is trying to behave abusively towards have relationships with those in the target community then this attack may not succeed as news of Rogue's abuse may get back to the target community. If this happens then Rogue will lose his membership in the target community. Conversely, greater impact can be achieved if the agent Rogue is abusive towards is an influential member of his (deceived) community, as the influential agent's negative recommendations about Rogue will have more weight.

This attack is not possible on identity-less communities. As such it is only applicable to structural communities.

Outsider Attack

In an outsider attack, Rogue joins communities external to the target community. Rogue's goal here is to influence the communities that he is a member in order to grow distrust of the target community. To do this Rogue will have to build a good reputation as a recommender. Once this is successful he can then influence the deceived groups by spreading negative reputation about the target community within the deceived communities.

An example of this attack can be seen in a recent attempt by the Recording Industry Association of America to flood the file sharing community with corrupt copies of music files [Shi03]. Coined the "Crush The Connectors" strategy by Clay Shirky, the RIAA attack exploits a fundamental property of systems that exhibit *small world* patterns: that communities are connected to each other through highly connected nodes [WS98]. The network of file-sharers is assumed to exhibit this property. Thus, by carefully identifying and then attacking these highly connected nodes, this strategy works well towards disconnecting the global community of file-sharers.

11.3 The Benign Agent

On the receiving end of an attack are the benign members of the community whose concerns are to just "get on with things" without exposing themselves to too much risk. The motivation of an agent for joining a community (where members exchange recommendations about a specific context) is to identify good agents to interact as well as bad agents to avoid.

Attacks on benign agents can be directly or indirectly mounted by Rogue.

11.3.1 Direct Attack

In a direct attack on the benign agent (whom we will refer to as Benign), Rogue joins Benign's community and then tries to decrease Benign's reputation within the community. To do this Rogue must first be able to build other members' trust in himself as a recommender to gain influence. Once this is achieved then Rogue can then start sending out negative recommendations about Benign. Rogue will continue to do this until such time as the community's members have lost trust in Benign and this effectively purges Benign from the community.

To achieve this Rogue must determine the network structure of the community by discovering the relationships between Benign and his trusters and trustees. Rogue must then reduce the trusters' trust in Benign, while at the same time minimising Benign's awareness of this attack being executed (to prevent Benign counter-attacking Rogue, especially if Benign is quite influential in the community). For the latter (minimising attack awareness) Rogue can make use of Benign's trustees – if Rogue can get the trustees to tell Benign that everything is normal then this will blind Benign to the attack being mounted.

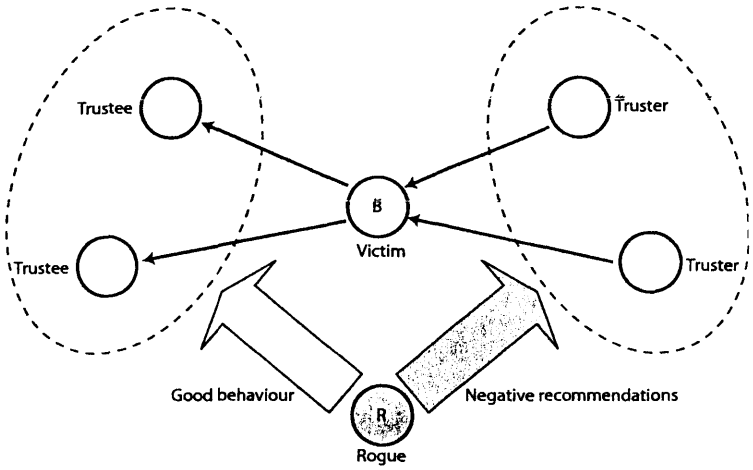


Figure 11.6: Direct attack on benign agent's reputation by spreading negative reputation about B to his trusters while showing good behaviour to his trustees to hide the attack from B.

The negative collusion attack, as described on page 217 can also be considered a direct attack on Benign as the target is an individual agent.

11.3.2 Indirect Attacks

An indirect attack is simply an attack on the community of which Benign is a member. The idea is to lower Benign's reputation through his affiliation with a community with low reputation. This attack may be useful when Rogue wants to reduce Benign's influence on other communities but is not able to directly attack Benign (because of the influence Benign has). An example of why Rogue may want to mount this attack could be because Benign is outspoken about the bad quality of a certain product that Rogue manufactures and Rogue wants to minimise the impact of that criticism.

The techniques for this attack are the same as discussed in §11.2.

11.4 The Malicious Agent

We will regard those agents that behave intentionally maliciously or exploitative as those that will need to be identified and discouraged or removed from a community. Malicious attacks are those intended to harm an agent or community directly, for example to reduce the community's trust in a competitor. Exploitative behaviours are those concerned with cheating and may involve malicious attacks if it aids exploitation.

The methods available to Rogue for mounting attacks have been discussed above. Here we will

briefly look at the costs incurred in carrying out an attack.

11.4.1 Cost of Attack

The higher Rogues reputation in the community, the better his potential to abuse the community becomes. However, each time Rogue carries out an attack or abuses the community, there will be a record of this abuse somewhere in the community and this will contribute to the deterioration of his reputation. This deterioration of his reputation is the cost of the attack.

If Rogue intends to remain in the community, then the cost of re-establishing the lost reputation will have to be less than the gain from abusing the community with this reputation for the attack to be a beneficial action.

If Rogue intends to leave the community after the attack, then there is no cost incurred.

11.4.2 Post-attack Strategies

Rogue must decide whether he intends to stay in the community or leave after an attack. If Rogue decides to leave and move to a new community then Rogue will need to ensure that he cannot be linked to his bad reputation in previous communities. Insisting on referrals from new agents is one way to deter Rogues that are on the run.

If Rogue intends to leave the community and return to it with a different pseudonym then the referral approach may also be a good deterrent. One way Rogue can overcome this is to manage a set of pseudonyms $R_1..R_n$ within the community and carry out a positive collusion attack (see §11.2.1) to maintain high reputation levels in them. Once a pseudonym R_1 has reached a sufficient reputation level, R_1 can carry out an attack and then leave the community. The rest of the pseudonyms, $R_2..R_n$, in the collusion set still remains in the community. A new pseudonym, R_{n+1}, can then be added to the set after R_1 drops out of the community – this will ensure there are sufficient pseudonyms to keep building reputations for each other and carry out repeated attacks. Rogue may also need to employ a Sybil attack approach to ensuring that other Benign agents are not able to tell that the pseudonyms belonging to Rogue all belong to the same attacker.

Continual abuse of a community will change the dynamics of the community in such a way that will make it difficult for Rogue to abuse it in future by keeping to the same strategy (see §11.7 above for this discussion). Therefore Rogue will need to find an optimum level of abuse that will still benefit Rogue while keeping the community stable so that he does not lose any investment in his continuous attack strategy.

A Rogue who intends to remain in the community after a malicious attack (personal attack on an agent) may try to build influence in the community. This will allow him to counter any bad opinions about Rogue that the attack victim may try and spread about Rogue. For example, Rogue may try to make the other community members believe that the victim is a liar and then attack the victim. When the victim then tries to tell others that Rogue is a malicious agent, his word will have less weight than Rogues.

We now look at two other important issues; the notion of *influence* of an agent on other agents,

and the question of whether it is possible for a community's trust topology to be discovered by an attacker.

11.5 Influence

An agent *A* is said to be more influential than another agent *B* if recommendations given by *A* have a higher probability to be considered and to be more persuasive in the community than *B*'s. For example, if after a political debate, a public opinion poll shows a higher percentage of support for the first politician (and if we assume that this support correlates with belief in the politician) then the first politician is said to be more influential than the second, in that debate.

The strength of an agent's influence in the community depends on three factors:

- The number of agents who trust that agent – if more people trust you than the next agent then you will be able to influence more people.
- The level of trust in each of those trust relationships – if someone trusts you more than the next agent then you will be able to influence him more.
- The level of influence that agent's trustees have – if those you influence can influence others then your influence is multiplied.

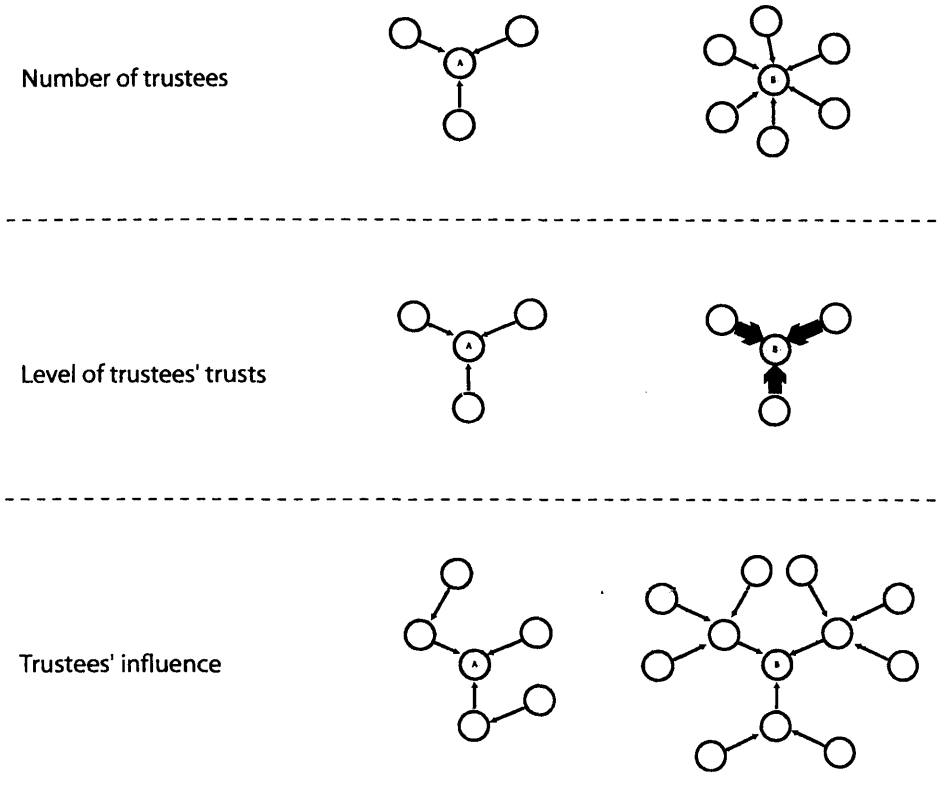


Figure 11.7: B is more influential than A because B may have more trustees (top), or may have higher trust from his trustees (middle) or may have a higher number of trustees who are themselves influential (bottom).

Influence is an important concept for Rogue because, firstly, he will need to use his influence to try and reduce trust in a relationship through his recommendations to the truster. Secondly, for efficient attacks, Rogue will favour deceiving the more influential members of the community so that they can in turn influence others, thereby allowing Rogue to spread rumours through trusted channels throughout the community.

This also means that it is in the interest of a community member in an influential position to protect himself from being influenced by a Rogue. If not and when it is discovered that he has given erroneous recommendations, his credibility will come into question and he may lose his influence. Therefore, the more influential an agent becomes, the more he has to be careful about giving opinion on others, if he wants to maintain that influential position.

Howison [How03] discusses influence briefly in his literature survey, and to our knowledge his is the only work to explicitly address influence in reputation systems.

11.6 Community Topology Visibility

Some of the attacks discussed above requires some amount of knowledge about the community's topology of trust relationships as a pre-requisite. Rogue needs to know which relationships to attack in order to break up the community, and he will need to know which agents to influence, which requires knowledge of the influential agent's trustees. He will also need to know his own level of influence so that he knows when he has done enough and is ready to mount the attack.

Details of how Rogue may go about discovering the topology is beyond the scope of this work as we suspect the actual methods vary from one implementation to the next. One example of how Rogue may discover the topology in a purely distributed reputation system is by broadcasting requests for recommendations throughout the network and build up a picture of the topology 'graph' by piecing together individual recommendation 'edges'.

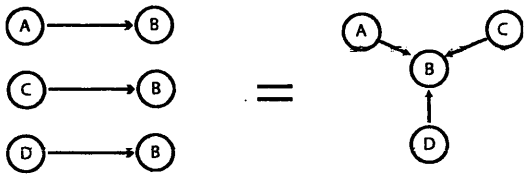


Figure 11.8: Individual recommendations can be used to build up a picture of the community trust topology.

This hints at an added risk of actually giving out recommendations because of the possibility of topology discovery. Furthermore, recommendations are potentially sensitive pieces of information as they tell about one party's trust, or distrust, in another. Minimising the amount of recommendations given to a malicious agent (or any agent for that matter) will be difficult to do without central control, and even if it is possible to control release of information then a successful Sybil attack[Dou02] by Rogue can subvert this control.

As such, a recommender may have to determine a recommendation requester's trustworthiness and motivations before returning a recommendation. Or alternatively, some measure of trust in the community as a whole can be used to determine release of recommendations. For example, if the rec-

ommender trusts that the community adheres to strict membership policy and that members are all benign then recommendations may be more freely given. However, this is an additional step in the recommendation protocol and may incur additional costs and change the dynamics of the community – benefits may be more expensive to receive.

We discuss dynamics of the community and its members next.

11.7 Post Attack Reactions and Dynamics

The action of attack will provoke a reaction by those attacked. The reaction of a benign agent after an attack, whether directly on himself or the community, can be grouped into two general cases:

- To *remain* in the community and continue to be affiliated to it. Reasons for this could be because membership still brings its advantages, because of group loyalty or because the agent is going to mount a counter-attack.
- To *leave* the community, either voluntarily or involuntarily. A motivation example of voluntary withdrawal may include perceiving that the cost of membership to exceed its benefits, losing trust in the community as a result of a successful attack and minimising or avoiding further attacks. Involuntary withdrawal may be the result of a successful attack whereby the victim gains a negative reputation in the community and is consequently evicted by the community members.

There are many factors that may motivate either of the reactions above. These motivations will depend on the perceptions and motivations of the attack victim and other members of the community. A complete analysis of these factors is beyond the scope of this work. Collaborative work with those within the social science arena will be required for this purpose. Below we outline our assumptions on a few possible scenarios post-attack.

Perceiving that membership may still provide benefits, the victim may decide to remain in the community. However, the victim's perception of the risks of membership would have changed (it is now higher) and as such the victim may begin to be more selective about whom he interacts with. This example can be observed in financial markets and was highlighted in a recent speech by Alan Greenspan, Chairman of the US Federal Reserve [Gre04]:

> *After the revelations of corporate malfeasance, the market punished the stock prices of those corporations whose behaviours had cast doubt on the reliability of their reputations.*

This means that the victim will now have fewer strong trust relationships. If Rogue remains in the community and continues to claim more victims then the overall outcome would be the weakening of the community as each victim reduces their number of trust relationships. Re-strengthening the community will take longer than it would if the community was starting afresh. This is because its members now will have to start building trust again after losing trust and this is harder to do, as discussed in §2.7.1.

One way of speeding up the strengthening of the community is by identifying Rogue and purging him from the network and making this publicly known. This will allow the community members to lower their perceptions of risk and lower their trustworthiness threshold for interaction.

Leaving a community may split the community as a result. Consider the community in Figure 11.9 which B leaves. Bs leaving effectively splits the community into three parts, one consisting of A and D, the other of C, E and F, and the third of solely B (Figure 11.10. B can then build a new community now by gaining the trust of G and H. In the example, G may not be in any community but H is already part of a community that includes also J (Figure 11.11).

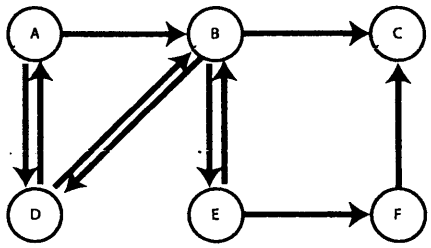


Figure 11.9: Original community.

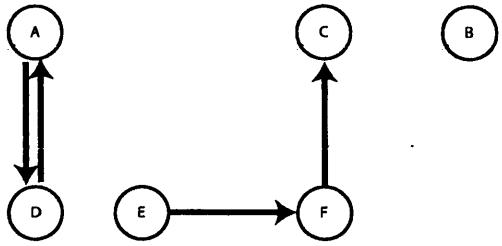


Figure 11.10: B leaves community.

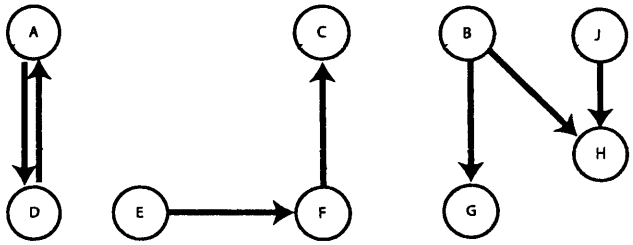


Figure 11.11: B forms new community with H and J.

For structural communities, an agent can explicitly leave the community because there is a tangible structure that the agent can disassociate with. In an emergent community, all the agent has is a local view of the community. For example, in the least diagram above of the community consisting of B, G, H and J, Bs view of the "community" is simply his relationships with G and H. He is not aware of the existence of J, its relationship with H nor whether any relationship exists between G and H. Thus in an emergent community, "membership" of the community is also an emergent notion. Removal from a community emerges from the total reconnection of all local relationships between the agent and the community members.

However, an attacker whose goal is to purge an agent from the community may, through network analysis, be able to determine the networks topology and through updating this topology may be

able to monitor the attacked agents gradual reduction of the number of active relationships between him and the community. Thus the attacker can discern when the victim is finally purged from the community.

11.8 Chapter Summary

In this chapter we have looked at possible threats to a reputation system. This was carried out by analysing three abstractions from these systems; the benign agent, the malicious agent and the community within which they interact.

We assumed that all agents interact within a community of agents, all participating towards a common goal or task. A community can be defined structurally or exist as an emergent phenomena, based on similarity of the context of interaction by agents.

A malicious agent may attack a community of agents. We defined three classes of attacks on communities; exploitation, weakening and destruction. Several different examples of attacks in each of these classes were given.

A malicious agent may also attack a benign agent, either directly or indirectly. Indirect attacks are merely attacks on the communities within which the benign agent interacts.

We then looked at the dynamics of agents and communities after an attack. For the malicious agent, the cost of an attack is the cost of rebuilding his reputation prior to mounting another attack. Thus his post-attack strategies will influence his attack strategies. This depends on whether he intends to stay in the community and maintain/rebuild this reputation, or leave after the attack.

We also briefly discussed how a community may break apart from an attack.

In order to attack a reputation system successfully, the malicious agent must be influential and must be able to influence other influential agents in the community. Thus we also looked at the concept of influence and the visibility of community topology.

Chapter 12

Future Work

We must learn our limits. We are all something, but none of us are everything.

Blaise Pascal

The work presented in this dissertation is an early attempt at modelling trust for computer systems based on social trust properties. This research has opened up many areas for investigation. We highlight areas which we think will make for interesting and fruitful work. They can be divided into five general categories:

1. Analysis of the Ntropi model
2. Implementations of Ntropi
3. Refinement of the social model
4. Refinement of the computational model
5. Rating strategies

12.1 Model Analysis

There have been a number of simulations in the literature that analyse properties of trust models and reputation systems. The primary contributions of these efforts to date was to show that reputation systems using simplified trust models can increase the robustness of communities of agents against malicious or anti-social behaviour [YS02, JHF03]. Furthermore, game theory has given us insight into how co-operation may evolve amongst rational agents [Axe84]. However, limitations in the models analysed, such as whether rationality is a realistic assumption of how agents behave in the real world, signal the readiness of the research community to augment the important findings above with analyses of systems with more complex models of agent behaviour.

In this work, we have attempted to contribute to this by describing an instance of how strategies in social trust can be modeled in computer systems. We believe this will be beneficial in informing future work on simulating intelligent agents that are able to evolve strategies according to the environment these agents find themselves in or other agents they are targeting. Future work in this

area will require a much more sophisticated model of trust reasoning in agents, which we feel will complement the simplified models in current efforts that are better suited to understanding the role of trust models in various communities.

It is thus also timely that the threat analysis presented in this chapter is available as more sophisticated platforms for trust analysis currently being developed. One such example of the latter is the proposed Trust Competition Testbed [tru] first mooted in the AAMAS 2004 Workshop on Trust in Agent Societies, and whose prototype is due to be demonstrated in the annual AAMAS conference in 2005. We believe collaboration with this trust testbed will be useful.

Further general properties of the model will also benefit from analysis, particularly on how well this model scales and the rate of convergence in reputation and actual agent behaviour in a community.

12.2 Implementation

There are implementation questions that will also need to be addressed in future work. In particular, we would be interested in the following:

Context How can contexts be represented? For recommendations to be successfully exchanged, the exchanging parties must know which contexts the recommendations are being requested for. If contexts can be independently defined by agents then some form of translation between contexts will be required. This places significant overheads on the establishment of recommender relationships as the identifier and semantics of each context will have to be learnt. This may be less of a problem if used in a closed network with a single network management authority, such as the example of using RDFs [(ed04] for the exchange of medical data. However this issue becomes significant in open networks. An alternative would be to centralise the definition of contexts. This may reduce the need to "learn" about new contexts, but this carries with it the usual problems of centralised approaches, particularly the need to capture heterogeneity in the semantics of a particular context within a single, possible hierarchical, ontology.

Trust level scale As we saw in our survey of current trust models (Table 4.7, page 97), there are many ways that trust value scales can be, and indeed have been, defined. This reflects the many different perceptions of how different levels of trust, and its graininess, should be represented. A useful trust scale is the subject of future work. A hypothesis is that different trust scales will be demanded by users of different applications, due to the need to pin down the semantics of each trust level to the underlying policies of the application domain. If this is so, and if there is a need by the user to carry trust values across domains, then some form of translation mechanism between trust scales may have to be investigated.

User interface One of the least investigated areas in trust research today is how users will interact with a trust management system. Early systems have ranged from programming-like interfaces, such as the AWK language interface of KeyNote [BFI+99] and Prolog interface of SULTAN [Gra03] to simple numerical ratings mechanisms, such as those used in eBay [eba]. The simplicity of the user interface in these systems reflects the sophistication of the underlying trust model. For example, KeyNote allows flexible and automatic policy enforcement based on public keys, to be used by experts, while eBay is simply a system that aggregates

user ratings, to be used by any user. With the multitude of customisable parameters in Ntropi, research into the best method to present the customisation options to the user will be required.

Resource requirements Ntropi uses a range of databases for data used in evaluating trustworthiness levels and reputation. Additionally, a simple database maintenance model have been designed which act as a limited memory management tool. An investigation into actual resource requirements of Ntropi, in terms of memory usage and network bandwidth consumption, will be required to so that implementers can work out exact requirements.

Interface with trust managers Throughout this dissertation, we worked with example trust management policies to illustrate the use of Ntropi and to drive the design of its interfaces. The example external policies were based on the concept implemented by Keynote, which is a freeform approach to policy specification using a high level language. Actual implementations which interface with actual trust managers in future work will help refine the interface requirements and provide a realistic picture of how an reputation calculation engine such as Ntropi can work with policy enforcing systems such as the trust managers above.

12.3 Social Model

Context experience transference Ntropi treats each context as distinct. However, as human beings, we sometimes re-use experiences with a prospect in one context to make judgements about the prospect's trustworthiness in another. For example, if Bob has a stable trust relationship with Alice in the context for 'being punctual', then perhaps Bob can use this experience and transfer it to another context for 'keeping promises' with Alice. The ability to do this presumes some form of inter-relationship between the contexts between which experiences are to be transferred.

One way to represent this would be for the user to specify which categories are related. This relationship can then be used to transfer experiences between the related contexts. An improvement on this method would be to specify degrees of relevance between contexts so that multiple related contexts can be weighed according to how relevant they are to the context in question.

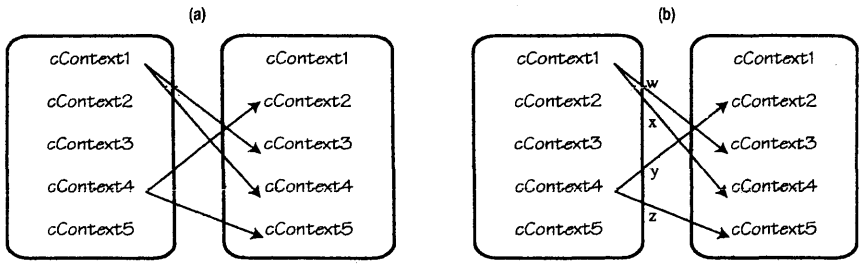


Figure 12.1: Contexts can be inter-related so that experiences from one (a). Degrees of relevance can be used where more than one contexts are related, so that the correct weighting can be given to the contexts on the right based on how relevant they are to the context on the left (b).

The methods above must be manually executed, and as such may place limits on the autonomy of agents and may prove to be a tedious exercise for users. A fully automated example may be

desirable in this case. One example would be for the agent to build correlations by analysing the distribution of all experiences with all past trustees in different contexts. Contexts that display similar experience distributions may be related so that experiences can be transferred between them for future evaluations of prospects that are known only in one of those contexts. A more intelligent strategy would be for the agent itself to place degrees of relevance on the context of inter-relationships such as above, based on measures of closeness between distributions of different contexts.

An additional issue with context relationships is that experiences are also now inter-related through the inter-relationship of contexts for which they have been asserted for. Thus, procedures such as purging of old experiences must now take into account whether these experiences may still be needed for the evaluation of other contexts. Thus this may increase memory requirements.

On the other hand, identifying opportunities for sharing experiences between contexts may also help reduce memory requirements. If two experience records, one for each inter-related context, are similar in rating, then only one of them need be stored as the single record can be used for evaluation of either of the inter-related contexts.

Forgiveness Currently, a transition into the untrusted phase is effectively a one-way street. To allow an untrusted phase agent to be trusted again will require manual change of phase by a user. Future work will look at 'forgiveness' strategies by trusters so that, for applications that require it, untrusted agents can be re-considered for trusting again.

Consensus decision making We have not yet investigated how agents may use computational trust models to make joint decisions on trust. There are many instances where this may be desirable, for example where a company policy states that at least two signatures will be required to issue cheques over a certain amount. Further investigation into this facility will be required. As it stands, Ntropi is not able to be used for this purpose.

Self Trust One of the fundamental types of trust in the social science literature is trust in oneself, or self-confidence [Mis96]. In Ntropi, there are areas where an awareness of self-trust by an agent may be useful. One example would be when comparing personal experience with the recommendation of others. If an agent is a new participant in a community, he may not yet have enough experience to judge the trustworthiness of others with confidence. In this case his self-trust with respect to ratings in this context would be low. Recognising this, Ntropi can thus elect to base trustworthiness measurements on recommendations instead of direct experience. Once a minimum threshold of direct experience is reached and self-trust reaches a minimum required level, then Ntropi can switch to using direct experiences to base trustworthiness evaluations on. Further investigation into the dynamics of self-trust and how it can be used to complement existing strategies in Ntropi will be a very useful exercise.

Decision making theory Models for trust and reputation evaluation can be seen as tools to support decisions making systems. The output provided by Ntropi, and other such models, are used by decision systems (or expert systems) to select the most appropriate strategy to execute. Thus, a better understanding of expert systems and theories on how agents make decisions will help direct future development of Ntropi, particularly on the required inputs and outputs if it is to interface with such expert systems.

12.4 Computational Model

Further investigation into alternatives to the computational model used in Ntropi will be the subject of future work. Three primary areas are the subject of this focus.

Evolving strategies Various parameters in the Ntropi policy are values that must be manually specified at the bootstrapping phase of each instance of the model. The weightings for the impact of different experience levels (Stable Phase Experience Weights, page 160) in for stable phase relationships is one example. However, the learning process involved in the early life of agents, together with the dynamics of trust relationships and the environment within which an agent finds itself, may necessitate modifications to these policy parameters. If an agent is to be truly autonomous then he has to learn to evolve these policies and evaluation strategies in order to adapt.

On such example could be in the selection of classifiers in the evaluation strategy for Unfamiliar agents (page 143). Currently, the user specifies which classifier to use for each context. However, an intelligent agent should be able, from experience, to select the classifier that best predicts experiences within that context.

Summarisation The summarisation functions used by Ntropi on past experiences are simple statistical measures of the median for central tendency and the semi-interquartile range for measures of dispersion. These were selected based on statistical principles on measurements on qualitative data [Jai91]. As with any statistical measure, tests carried out on real world samples will tell us whether it is a realistic measure.

Intelligent RRQs At the moment, recommendation requests are sent to recommenders based on how much the requester trusts the recommender for a given context. This has the advantage that any recommendation received will originate from a trusted source. However, it does not guarantee that a recommendation can be found in the first place. An additional parameter to recommender selection, such as affiliation matching, may help direct a more effective search for recommendations. In the affiliation matching example, we can imagine that a recommender who is a colleague of the prospect (affiliated to the same employer) has a higher probability of having an opinion about the prospect than an arbitrary recommender. The work of Yu and Singh [YS03] is an early effort that looked into this problem.

Arbitrary chain lengths Our model of recommendation chains is currently fixed at specific chain lengths using the *rn* prefix. In certain applications, this may be restrictive and may benefit from simplification of how recommender trust contexts are defined. A context for trusting recommenders for arbitrary chain lengths can be envisaged, whereby an agent is trusted to find recommendations regardless of how many intermediate recommenders it required to get there. This is akin to trusting an agent to be a 'lead' to other sources of information - all the requester is interested in is finding a quality recommender, he does not care how this recommender is found. Nevertheless, the requester will need to start from a known recommender, and thus the one most likely in the truster's opinion, will lead to a quality recommender being asked. Another advantage for supporting this context would be faster and simpler recommendation evaluation as no recursive evaluation of intermediaries are involved. We are uncertain, however, on what this 'lead' context may entail in terms of how the semantics of the context

should be defined and how one evaluates an experience for this context. Lessons learnt from further investigation into this area may provide for simpler recommendation chain models.

12.5 Rating Strategies

In the design of Ntropi, we have delegated the task of actually rating experiences to a hypothetical external entity that carries out application-specific evaluations of experiences from interactions (see §5.3). The reason for excluding this rating function from the model is that how one would rate the experience from an interaction varies from one application to another and is highly subjective. An attempt to generalise an experience rating process across all possible application domains would be a whole research area in itself.

Nevertheless, future work into understanding the general nature of rating strategies available to agents in a community will help inform at least two areas: 1) an insight into implementation issues and the feasibility of engineering a general rating model that could be re-used in various application domains, 2) an insight into malicious agents who may attempt to subvert the trust model by evolving strategies that maximises received ratings.

Thus the basic question is this; how does a truster decide whether an observed behaviour is positive or negative in nature? At least three strategies are possible:

Contracts The question the truster will be asking during the rating phase in this case would be whether the trustee behaved as agreed prior to the interaction. For example, in our running example from previous chapters, Alice and Bob may agree, prior to Alice purchasing processor time from Bob's host, that her program may not attempt to make connections to any other host on the Internet. Thus, if Alice's code did attempt to make an external connection then she would have defaulted on her contract with Bob, and Bob can view this as a negative experience. There are two significant advantages to this approach.

Firstly, it gives Alice the opportunity to learn about the standard of Bob's expectations in users of his service and it allows Alice to decline the service if she feels that she is not able to meet the conditions in the contract.

Secondly, it allows for clearer separation of responsibilities, which simplifies ratings and contributes to less ambiguous recommendations. This simplification is due to the opportunity to define contexts in more concrete terms. In the example above, the negative behaviour may be related to the context "behaves as per contract" or "behaves as agreed". In a recommendation, this context is less ambiguous than, say, "behaves maliciously" which is open to interpretations on the semantics of "malicious".

The disadvantage of this approach is that contracts can add additional overheads to interactions. In certain applications this may be unacceptable. For example, in the routing protocols of ad hoc networks with highly dynamic topologies, where community membership is short lived, contract-based interaction may not be feasible.

Local expectations The other extreme is for the trustee to behave as it wishes and leave it up to the truster to rate the experience of the interaction based completely on the truster's local standards. This adds no overhead to interactions, but it adds uncertainty in two areas;

The first is related to the confidence of the truster as a rater. The truster may not want to be known as an unfair rater as this reputation may not be desirable for various reasons (it impedes progress in building influence in others, for example). This uncertainty is due to the subjectivity of "unfairness" – what is a fair rating to one agent may not be so for another. If the truster's confidence is high in his rating fairness then this may not be an issue. If not then this uncertainty may drive the truster to avoid ratings on the extreme values of the trust scale.

The second is related to the reputation of the trustee. If it is possible to determine how each agent rates behaviours of their trustees then it would be much easier for prospects to select the ones that will most likely provide a good rating, thus ensuring as high a reputation as possible for the prospect. If not, however, there may be costs involved in searching for a truster that can provide good ratings based on conditions that the prospect can meet.

Social norms The third strategy is for a truster to rate according to the rating standards of the community. In order to learn the community's standards some amount of time must be invested by a new member before he is able to rate others. Thus this may not be desirable if new community members are expected to make early rating contributions to the community. However, in our model, trust relationships go through an initial building (Fragile) phase anyway, so a learning period may also be necessary for other members to build trust in the new member. We identify three general strategies that the newcomer and other community members can employ to help him build a reputation as a good recommender.

The first is the *silent observation* strategy. The newcomer holds back making recommendations but makes silent ratings which she keeps to herself. At the same time she requests recommendations about the same prospect and context from other community members, and compares this with her own silent ratings. Over time she will be able to fine tune her own rating standards by comparing them with the recommendations she receives. Once she has reached a stage where her silent ratings are consistently similar to the other recommendations, i.e. the social norm in rating standard, he can then begin making reliable recommendations to others in the community.

The second is the *rating mentor* strategy. A new member would carry out ratings with the help of a trusted mentoring agent from whom the new rater will learn social rating standards. After an interaction, the newcomer would pass on details of the experience to the mentor and the mentor would then return a rating as though he himself had experienced that interaction. This allows the new rater to match ratings to actual experience data. Furthermore, the recommendation from the mentor, which is meant to help the newcomer learn, can also be usable by other community members, so that no experiences need go without usable recommendations. In a real application, the mentor may actually be a person which gives feedback to the software. This is very similar to learning algorithms in artificial intelligence – such as neural nets – where the software is provided with learning samples which it uses to fine tune internal recognition parameters.

The third is a *collaborative guidance* strategy. Here, a newcomer would transmit recommendations from early on. However, the recommendations would be marked as coming from a new community member, or 'learner recommender'. This allows the requester to take this 'learner' status into account when processing the recommendation and take the appropriate actions if necessary, such as giving it a lower weighting when combined with other recommendations. This 'learner status' may also prompt established members to reply to the newcomer with

his own opinion of the prospect so that the learner can match his rating to one from a more established member. It may also be used to invite other agents to be the newcomer's mentor(s).

12.6 Chapter Summary

In this chapter we have discussed several areas of interest for future research and a natural progression from the body of work presented in this dissertation. These areas can be broken down into the sections presented below.

Model analysis Further analysis of the model will be beneficial. In particular, the simulation of intelligent malicious agents that are able to evolve attack strategies, Ntropi's scalability and convergence of rating and actual behaviour.

Implementation Implementation exercises will highlight various usability aspects, such as the representation of contexts, the adequacy of the trust level scale and the human-computer interface for such an application. Furthermore, resource requirements and inter-operation with trust managers will have to be looked into.

Social model The social model underlying Ntropi can be extended to include several other aspects of social interaction, such as transferring experiences between related contexts, allowing Untrusted agents to be 'forgiven' and allowing groups of agents to derive a consensus on decisions. Additionally, a better understanding of decision-making systems, such as expert systems, will enable improvements to future versions of Ntropi.

Computational model The computational model presented in this work is one instance of specifying the properties of our desired model. Future work will look at the ability of agents to evolve policy strategies as this will greatly improve agents' adaptability compared to the current, static model. The properties of the simple statistical summarisation functions will also need to be analysed in light of the requirements of the model. Additionally, a more intelligent approach to broadcasting recommendation requests, with a goal of higher return rates, will be explored.

Rating strategies Experiences are rated externally to Ntropi. However a better understanding of choices for rating strategies will help inform implementations. In addition it will also help uncover possible strategies available to malicious agents for gaining high ratings.

Chapter 13

Conclusion

Recent developments in the pervasiveness and mobility of computer systems have invalidated traditional assumptions about trust in computer communications security. It is now increasingly difficult to guarantee certainty and control over all network communications, upon which traditional security models are built. One of the reasons for this is that the question of whether an entity should be trusted is one of the facets of the design of secure systems. In a fundamentally decentralised and open network such as the Internet, it is not possible to assert control artificially over this decision.

In view of the fact that online agents represent users' digital identities, we believe that it is reasonable to explore social models of trust as a basis for secure agent communication. We have demonstrated that it is feasible to design and formalise a dynamic model of trust for secure communications based on the properties of social trust, through the following contributions of this work:

Critical review of social trust properties. In order to be able to specify properties of social trust in a computational trust model we must first understand what the properties are, and the dynamics of their interactions. A review of the social science literature including sociology, psychology, economics, political science and philosophy were carried out with careful attention to those aspects of trust relevant to computer communication.

Rigorous analysis of existing models. The findings from the review of social trust were used to analyse existing computational trust models. We showed that while current models *do* contain some of the essential properties of trust, many lack the support for *how* these properties evolve over time, i.e. the dynamic aspects of trust.

Formal model specification. A careful design of a trust model, Ntropi, with the essential properties of trust and algorithms to support its dynamics was carried out. To ensure that Ntropi could be used to engineer real implementations, two aspects of the methodology were highlighted. Firstly, a generic trust management tool, based on existing systems such as Keynote [BFI+99], was used to determine how applications will interface with Ntropi. Secondly, to ensure unambiguity and future analysis of its properties, the well known specification language Z was used to specify it.

Threats analysis. The effectiveness of this model depends on the robustness of its implmentation and an understanding of its weaknesses. An analysis of possible strategies for subverting the

model was carried out. This contributed to an understanding of how an intelligent malicious agent may evolve strategies for subverting the model, and consequently, how attacks can be prevented.

In addition, novel algorithms for trust-reasoning have been contributed. These include trustworthiness measures based on generalised experiences and stereotyping, recognition of the different phases of a trust relationship and its feedback loop, trustworthiness of recommenders based on consistency of recommendation quality, learning to translate between different standards of rating based on 'semantic distances' between trust values, tighter policies on evaluation of recommendation chains and a simple measure of reliability of reputational information.

Finally, aspects of this work have been published in peer-reviewed conference proceedings and used in Sun's JXTA platform and Ericsson Research's trust model prototype.

Acknowledgements

The following people have been essential ingredients in the development of this dissertation:

Firstly, my supervisor, Dr. Stephen Hailes, whose generosity, patience and commitment is above and beyond the call of duty. Thank you for introducing me to this exciting subject, for your enthusiasm which motivated me when my own have been lacking and your inspirational approach to research. And for not giving up when I almost did! Words do not describe my gratitude.

Professor Steve Wilbur, Professor Jon Crowcroft and Professor David Rosenblum for invaluable input from the masters.

The people at UCL Computer Science, especially Dr Ian Brown, Dr Nadia Kausar, Dr Jungwon Kim, Dr Adil Qureshi, Dr Paul White, Dr Rafael Bordini, Giannis Koufakis and Mohamed Ahmed, for companionship at various levels.

Dr. Barbara Misztal at Griffith University, Australia, for the invaluable comments from an expert sociologist.

Tenaga Nasional Berhad, Malaysia, for the sponsorship and conference trip funding.

All my friends who constantly remind me of the bigger picture in life. You know who you are.

My parents, for the moral and financial support, which I couldn't have lived without. To Alwin, Alfiza and Alman, for their warmth and expert skill in making me smile during the coldness of solitary research work. And for your patience and belief in me in the past years. I love you all very much.

Bibliography

[2202] ISO Technical Committee JTC 1/SC 22. Z formal specification notation – syntax, type system and semantics, ISO/IEC 13568:2002. Technical report, ISO, July 2002.

[AD01] Karl Aberer and Zoran Despotovic. Managing trust in a peer-2-peer information system. In Henrique Paques, Ling Liu, and David Grossman, editors, *Proceedings of the Tenth International Conference on Information and Knowledge Management (CIKM01)*, pages 310–317. ACM Press, 2001.

[Adl94] Jonathan Adler. Testimoy, trust, knowing. *Journal of Philosophy*, XCI(1):264–275, Jan-Dec 1994.

[AH00] Eyton Adar and Bernardo Huberman. Free riding on Gnutella. *First Monday, www.firstmonday.dk*, 5(10), October 2000.

[AH02] Karl Aberer and Manfred Hauswirth. An overview on peer-to-peer information systems. In *Workshop on Distributed Data and Structures (WDAS-2002)*, 2002.

[AJB99] R. Albert, H. Jeong, and A.-L. Barabsi. Diameter of the world wide web. *Nature*, 401:130–131, September 1999.

[AKC+00] Ion Androutsopoulos, John Koutsias, Konstantinos V. Chandrinos, George Paliouras, and Constantine D. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. In G. Potamias, V. Moustakis, and M. van Someren, editors, *Proceedings of the workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning, Barcelona, Spain*, pages 9–17, 2000.

[AR97] Alfarez Abdul-Rahman. The PGP trust model. *EDI-Forum: The Journal of Electronic Commerce*, April 1997. http://www.cs.ucl.ac.uk/staff/F.AbdulRahman/docs/.

[ARH96] Alfarez Abdul-Rahman and Stephen Hailes. A distributed trust model. In *DIMACS Workshop on Trust Management in Networks, New Jersey, USA.*, October 1996.

[ARH97a] Alfarez Abdul-Rahman and Stephen Hailes. A distributed trust model. In *Proceedings of the New Security Paradigms Workshop 97, Great Langdale, Cumbria, UK*, 1997.

[ARH97b] Alfarez Abdul-Rahman and Stephen Hailes. Using recommendations for Managing trust in distributed systems. In *Proceedings IEEE Malaysia International Conference on Communication '97 (MICC'97), Kuala Lumpur, Malaysia*, November 1997.

[ARH99] Alfarez Abdul-Rahman and Stephen Hailes. Relying on trust to find reliable information. In *Proceedings of DWACOS'99, Baden-Baden, Germany.*, 1999.

[ARH00] Alfarez Abdul-Rahman and Stephen Hailes. Supporting trust in virtual communities. In *Proceedings of Hawaii International Conference on System Sciences 33, Maui, Hawaii*, January 2000.

[Arr74] K. Arrow. *The Limits of Organisation*. Norton, New York, 1974.

[ASL97] A. Adams, M. A. Sasse, and P. Lunt. Making passwords secure and usable. In *Proceedings of HCI97*, 1997.

[AT91] Martin Abadi and Mark Tuttle. A semantics for a logic of authentication. In *Proceedings of the Tenth ACM Symposium on Principles of Distributed Computing*, 1991.

[Axe84] Robert Axelrod. *The Evolution of Cooperation*. Basic Books Inc., New York, 1984.

[Bai85] Annette Baier. Trust and antitrust. *Ethics*, 96:231–260, 1985.

[BAN90] Michael Burrows, Martin Abadi, and Roger Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1), February 1990.

[Bar83] Bernard Barber. *The logic and limits of trust*. New Brunswick, 1983.

[BBK94] Thomas Beth, Malte Borchedring, and Birgit Klein. Valuation of trust in open networks. In *Proceedings of the European Symposium on Research in Computer Science (ESORICS)*, 1994.

[BF87] P. Brann and M. Foddy. Trust and the consumption of a deteriorating resource. *Journal of Conflict Resolution*, 31(4):615–630, December 1987.

[BFI+99] Matt Blaze, Joan Feigenbaum, John Ioannidis, , and Angelos D. Keromytis. The keynote trust-management system, version 2, rfc 2704. Technical report, Internet Society, September 1999.

[BFL96] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *Proceedings of the IEEE Conference on Security and Privacy*, 1996.

[BH91] Susan D. Boon and John G. Holmes. The dynamics of interpersonal trust: resolving uncertainty in the face of risk. In Robert A. Hinde and Jo Groebel, editors, *Cooperation and Prosocial Behaviour*. Cambridge University Press, 1991.

[Bla89] P.M. Blau. *Exchange and Power in Social Life*. Transaction Publishers, New Bruswick, NJ, 1989.

[blo] Blogger. http://blogger.com.

[Boy02] Josh Boyd. In community we trust: Online security communication at eBay. *Journal of Computer-Mediated Communication*, 7(3), April 2002. http://www.ascusc.org/jcmc/vol7/issue3/boyd.html.

[Bra97] Marc Branchaud. A survey of public-key infrastructures. Master's thesis, Department of Computer Science, McGill University, Montreal, 1997.

[Cen04] CERT Coordination Center. Cert/CC statistics 1988-2004. http://www.cert.org/stats/cert_stats.html, October 2004.

[CER96] CERT. Advisory CA-96-21: TCP SYN flooding and IP spoofing attacks. http://www.cert.org/advisories/CA-1996-21.html, September 1996.

[CFL+97] Y.-H. Chu, J. Feigenbaum, B. LaMacchia, P. Resnick, and M. Strauss. Referee: Trust management for web applications. *World Wide Web Journal*, 2:127–139, 1997.

[CH96] Bruce Christianson and William S. Harbison. Why isn't trust transitive? In *Proceedings of the Security Protocols International Workshop, University of Cambridge*, 1996.

[Cha81] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications ofthe ACM*, 24(2), February 1981.

[CL99] George Cvetkovich and Ragnar E Lø:fstedt, editors. *Social Trust and the Management of Risk*. Earthscan, 1999.

[CMH+02] Ian Clarke, Scott G Miller, Theodore W. Hong, Oskar Sandberg, and Brandon Wiley. Protecting free expression online with freenet. *IEEE Internet Computing*, 6(1):40–49, 2002.

[Coc97] William G. Cochran. *Sampling Techniques*. John Wiley & Sons, NY, 1997.

[CY01] Rita Chen and William Yeager. Poblano: A distributed trust model for peer-to-peer networks. Technical report, Sun Microsystems, August 2001. www.jxta.org/docs/trust.pdf.

[Dai] Wei Dai. Towards a theory of reputation. cypherpunks@toad.com: Tue, 21 Nov 1995 15:32:08 -0800. Cypherpunks discussion list.

[Das88] Partha Dasgupta. Trust as a commodity. In Diego Gambetta, editor, *Trust*. Basil Blackwell, 1988.

[Dav02] Paul Davidson. Categories of Artificial Ssocieties. In Jaime S. Sichman, Francois Bousquet, and Paul Davidsson, editors, *Proceedings of Multi-Agent-Based Simulation II, Bologna, Italy*, volume 2581 of *Lecture Notes in Artificial Intelligence*. Springer, 2002.

[Deu58] Morton Deutsch. Trust and suspicion. *Journal on Conflict Resolution*, 2:265–279, 1958.

[DH65] J. B. Dennis and E. C. Van Horn. Programming semantics for multiprogrammed computations. Technical Report MIT/LCS/TR-23, MIT, 1965.

[Din00] Roger Dingledine. The Free Haven Project. Master's thesis, MIT, 2000.

[Dou02] John R. Douceur. The sybil attack. In *Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS '02), March 2002, Cambridge, MA, USA.*, 2002.

[drs00] drscholl@users.sourceforge.net. Napster protocol specification (OpenNap). http://opennap.sourceforge.net/napster.txt, April 2000.

[DS02] R. Dingledine and P. Syverson. Reliable MIX cascade networks through reputation. In *Proceedings of Financial Cryptography*, 2002. .

[DT02] Bicarregui J Dimitrakos T., Matthews B. Building trust on the grid. *Ercim News, Special issue on security.*, 49, 2002.

[E+99] Carl Ellison et al. SPKI certificate theory - IETF RFC 2693. Technical report, The Internet Society, September 1999.

[eba] eBay online auctions. http://www.ebay.com.

[(ed04] Dave Becket (editor). RDF/XML syntax specification (revised). Technical report, World Wide Web Consortium, February 2004.

[Ess97] Daniel Essin. Patterns of trust and policy. In *Proceedings of the New Security Paradigms Workshop 97, Great Langdale, Cumbria, UK*, September 1997.

[F+03] Linda Foley et al. Identity theft, the aftermath 2003: A comprehensive study to understand the impact of identity theft on known victims. Technical report, Identity Theft Resource Center. http://www.idtheftcenter.org, September 2003.

[FHMV96] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. Common knowledge revisited. In Yoav Shoham, editor, *Theoretical Aspects of Rationality and Knowledge: Proceedings of the Sixth Conference (TARK 1996)*, pages 283–258. Morgan Kaufmann, San Francisco, 1996.

[FK99] Ian Foster and Carl Kesselman. *The Grid.* Morgan Kauffman, 1999.

[FR01] Eric Friedman and Paul Resnick. The social cost of cheap pseudonyms. *Journal of Economics and Management Strategy*, 10(2):173–199, 2001.

[frea] Free haven. http://www.freehaven.net.

[freb] The freenet project website. www.freenetproject.org/.

[fri] Friendster. http://www.friendster.com.

[Gam88a] Diego Gambetta. Can we trust trust? In Diego Gambetta, editor, *Trust.* Basil Blackwell, 1988.

[Gam88b] Diego Gambetta, editor. *Trust: making and breaking cooperative relations.* Basil Blackwell, 1988.

[GDS+03] Krishna P. Gummadi, Richard J. Dunn, Stefan Saroiu, Steven D. Gribble, Henry M. Levy, and John Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *19th ACM Symposium on Operating Systems Principles (SOSP-19) Bolton Landing, NY*, 2003.

[Ger98] Ed Gerck. Towards real-world models of trust: Reliance on received information. *Meta-Certificate Group Publications, http://www.mcg.org.br*, 1998.

[Gid90] Anthony Giddens. *Modernity and Self-Identity.* Polity Press, 1990.

[Gin00] H Gintis. *Game Theory Evolving.* Cambridge University Press, 2000.

[gnu] Gnutella protocol development. http://http://rfc-gnutella.sourceforge.net/.

[GNY90] Li Gong, Roger Needham, and Raphael Yahalom. Reasoning about belief in cryptographic protocols. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1990.

[Gon93] Li Gong. Increasing availability and security of an authentication service. *IEEE Journal on Selected Areas in communications*, 11(5), June 1993.

[Goo88] David Good. Individuals, interpersonal relatios, and trust. In Diego Gambetta, editor, *Trust*. Basil Blackwell, 1988.

[Gra03] Tyrone Grandison. *Trust Management for Internet Applications*. PhD thesis, Imperial College London, 2003.

[Gre04] Alan Greenspan. Reputation, not new rules, must govern corporations. The Independent, Monday, 26 April, 2004.

[GRS96] David Goldschlag, Michael Reed, and Paul Syverson. Hiding routing information. In *Proceedings of the Workshop on Information Hiding, Cambridge, UK*, May 1996.

[GS96] Simson Garfinkel and Gene Spafford. *Practical Unix & Internet Security*. O'Reilly, 1996.

[Har] Jason Harris. Keyanalyze. http://dtype.org/keyanalyze/.

[Har88] Keith Hart. Kinship, Contract, and Trust: the Economic Organization of Migrants in an African City Slump. In Diego Gambetta, editor, *Trust*. Basil Blackwell, 1988.

[Har93] Russell Hardin. The street-level epistemology of trust. *Politics and Society*, 21(4):505–529, December 1993.

[Har96] Russell Hardin. Trustworthiness. *Ethics*, 107:26–42, October 1996.

[Hel67] Virginia Held. On the meaning of trust. *Ethics*, 78:156–159, October 1967.

[Hob47] Thomas Hobbes. *Leviathan*. E. P. Dutton & Co., New York, 1947.

[Hol99] Bengt Holmstrom. Managerial incentive problems: A dynamic perspective. *Review of Economic Studies*, 66(1):169–182, January 1999. http://ideas.repec.org/a/bla/restud/v66y1999i1p169-82.html.

[How03] James Howison. "an introduction to the literature on online reputation systems for the mmapps project". http://wirelessgrids.net/RepLitIntro/, August 2003.

[ITU97] ITU-T. *ITU-T Recommendation X.509, The Directory: Authentication Framework*, June 1997.

[Jai91] Raj Jain. *The art of computer systems performance analysis*. John Wiley, New York, 1991.

[JH94] K.M. Jackson and Jan Hruska, editors. *Computer Security Reference Book*. Butterworth-Heinemann, new ed edition, February 1994.

[JHF03] A. Jøsang, S. Hird, and E. Faccer. Simulating the effect of reputation systems on e-markets. In *Proceedings of the First International Conference on Trust Management, Crete*, May 2003.

[JK98] Audun Jøsang and S.J. Knapskog. A metric for trusted systems. In *Twenty First National Security Conference, NSA*, 1998.

[JM97] Steve Jones and Steve MArsh. Human-computer-human interaction: Trust in CSCW. *SIGCHI Bulletin*, 29(3), 1997.

[Jon96] Karen Jones. Trust as an affective attitude. *Ethics*, 107:4–25, 1996.

[Jøs96] Audun Jøsang. The right type of trust for distributed systems. In *New Security Paradigms Workshop*, 1996.

[Jøs97a] Audun Jøsang. Artificial reasoning with subjective logic. In *Second Australian Workshop on Commonsense Reasoning*, 1997.

[Jøs97b] Audun Jøsang. A model for trust in security systems. In *Second Nordic Workshop on Secure Computer Systems*, 1997.

[Jøs97c] Audun Jøsang. Prospectives for modelling trust in information security. In *Australian Conference on Information Security and Privacy*, 1997.

[Jøs98a] Audun Jøsang. Reliability analysis with uncertain probabilities. In *Fourth International Conference on Probabilistic Safety Assessment and Management, New York*, 1998.

[Jøs98b] Audun Jøsang. A subjective metric of authentication. In *European Symposium on Research in Computer Science (ESORICS)*, 1998.

[Jøs99] Audun Jøsang. An algebra for assessing trust in certification chains. In *Network and Distributed Systems Security*, 1999.

[Jos02] Samuel Joseph. Neurogrid: Semantically routing queries in peer-to-peer networks. In *International Workshop on Peer-to-Peer Computing (co-located with Networking 2002), Pisa, Italy*, May 2002.

[JS93] Hasan Jamil and Fereidoon Sadri. Trusting an information agent. In *International Workshop on Rough Sets and Knowledge Discovery*, 1993.

[JS94] Hasan Jamil and Fereidoon Sadri. Recognizing credible experts in inaccurate databases. In *Eight International Syposium on Methodologies for Intelligent Systems (ISMIS)*, 1994.

[jxt] JXTA website. www.jxta.org.

[KC93] Paul Krause and Dominic Clark. *Representing Uncertain Knowledge: An Artificial Intelligence Approach*. Intellect Books, 1993.

[Ken93] Stephen Kent. Privacy enhancement for internet electronic mail: Part ii: Certificate-based key management, rfc 1422. Technical report, Internet Society, February 1993.

[Kha96] Rohit Khare. Using pics labels for trust management. DIMACS Workshop on Trust Management in Networks, South Plainfield, N.J., September 1996. http://dimacs.rutgers.edu/Workshops/Management/Khare.html.

[Kha99] Rohit Khare. What's in a name? trust: Internet-scale namespaces, part ii. *IEEE Internet Computing*, 3(6):80–84, November 1999.

[Kly04] Graham Klyne. iTrust survey, presented at the 3rd iTrust working group internal meeting. http://www.ninebynine.org/iTrust/iTrustSurvey.ppt, October 2004.

[KMRT96] Tim Krauskopf, Jim Miller, Paul Resnick, and Win Treese. Pics label distribution label syntax and communication protocols, version 1.1, w3c recommendation 31-october-96. Technical report, World Wide Web Consortium, October 1996.

[KPC] Stephen Kent and Tim Polk (Chairs). Public-key infrastructure (x.509) working group. http://www.ietf.org/html.charters/pkix-charter.html.

[KSGM03] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Twelfth International World Wide Web Conference 2003, May 20-24, 2003, Budapest, Hungary*, New York, 2003. ACM Press.

[lin] LinkedIn. http://www.linkedin.com.

[LL92] Seymour Lipschutz and Marc Lipson. *2000 Solved Problems in Discrete Mathematics.* Schaum's Series. McGraw-Hill, 1992.

[Lor88] Edward H. Lorenz. Neither friends nor strangers: Informal networks of subcontracting in french industry. In Diego Gambetta, editor, *Trust*. Basil Blackwell, 1988.

[LSY03] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1), Jan/Feb 2003. Online version: http://dsonline.computer.org/0301/d/w1lind.htm.

[Luh79] Niklas Luhmann. *Trust and Power.* Wiley, 1979.

[Luh88] Niklas Luhmann. Familiarity, confidence, trust: Problems and alternatives. In Diego Gambetta, editor, *Trust*. Basil Blackwell, 1988.

[MA04] P Massa and P Avesani. Trust-aware collaborative filtering for recommender systems. In *Proceedings of International Conference on Cooperative Information Systems (CoopIS)*, 2004.

[Mar94a] Stephen Marsh. *Formalising Trust as a Computational Concept.* PhD thesis, Department of Computing Sciece and Mathematics, University of Sterling, 1994.

[Mar94b] Stephen Marsh. Optimism and pessimism in trust. In *Ibeoamerican Conference on Artificial Intelligence*, 1994.

[Mau96] Ueli Maurer. Modelling a public-key infrastructure. In *European Symposium on Research in Computer Science (ESORICS)*, 1996.

[MB04] P. Massa and B. Bhattacharjee. Using trust in recommender systems: an experimental analysis. In *Proceedings of iTrust2004*, 2004.

[MC96] D. Harrison MgKnight and Norman L. Chervany. The meanings of trust. Technical Report 94-04, Carlson School of Management, University of Minnesota, 1996.

[MCC95] D.H. McKnight, Larry L. Cummings, and Norman L. Chervany. Trust formation in new organizational relationships. In *Information and Decision Sciences Workshop, University of Minnesota*, 1995.

[MGM03] Sergio Marti and Hector Garcia-Molina. Identity crisis: Anonymity vs. reputation in P2P systems. In *IEEE 3rd International Conference on Peer-to-Peer Computing (P2P 2003)*, September 2003.

[MGM04] Sergio Marti and Hector Garcia-Molina. Limited reputation sharing in P2P systems. In *Proceedings of the 5th ACM conference on Electronic commerce (EC-04)*, pages 91–101, New York, May 1–8 2004. ACM Press.

[Mic04] Sun Microsystems. JXTA technology: Creating connected communities. www.jxta.org/project/www/docs/JXTA-Exec-Brief.pdf, January 2004.

[Mis96] Barbara Misztal. *Trust in Modern Societies*. Polity Press, 1996.

[mix] Mixmaster. http://mixmaster.sourceforge.net/.

[Moc87] P Mockapetris. Domain names - implementation and specification. *IETF RFC 1035, USC/Information Sciences*, November 1987.

[MvN47] Oskar Morgernstern and John von Neumann. *The Theory of Games and Economic Behaviour*. Princeton University Press, 1947.

[Mye97] Roger B. Myerson. *Game Theory: Analysis of Conflict*. Harvard University Press, 1997.

[Neu92] Peter G. Neumann. Where to place trust. *Communications of the ACM*, 35(10), 1992.

[Ope] OpenPrivacy. Sierra: An OpenPrivacy reputation management framework. http://sierra.openprivacy.org.

[Ope03] OpenPrivacy.org. Threat models. http://www.peerfear.org/fionna/index.html, 2003.

[PST96] Ben Potter, Jane Sinclair, and David Till. *An Introduction to Formal Specification and Z*. International Series in Computer Science. Prentice Hall, 2nd edition, 1996.

[Put95] Robert D. Putnam. Bowling alone: America's declining social capital. *Journal of Democracy*, 6:65–78, 1995.

[PW97] Charles Petrie and Meredith Wiggins. Pattie maes on software agents: Humanizing the global computer. *IEEE Internet Computing*, pages 10–19, July 1997.

[QL04] K. Quinn and A. Leddy. Ericsson research (internal) technical report ebs/cd/a-04:000737 uen. Technical report, Ericsson Research, August 2004.

[QOD+05] K. Quinn, O'Sullivan, D., D. Lewis, R. Brennan, and V.P. Wade. deeptrust management application for discovery, selection, and composition of trustworthy services. In *Proceedings of IDIP/IEEE 9th International Symposium on Integrated Network Management (IM 2005), Nice, France*, 2005.

[RA99] Cheskin Research and Studio Archetype. E-commerce trust study. www.cheskin.com, 1999.

[Ran88] P. Venkat Rangan. An axiomatic basis of trust in distributed systems. In *IEEE Symposium on Security and Privacy*, 1988.

[Ras96] Lars Rasmusson. Socially controlled global agent systems. Master's thesis, Kungl Tekniska Högskolan, Stockholm, 1996.

[Rea96] Joseph M. Reagle, Jr. Trust in a cryptographic economy and digital security deposits: Protocols and policies. Master's thesis, Massachusetts Institute of Technology, 1996.

[Rid97] Matt Ridley. *The Origins of Virtue*. Penguin Books, 1997.

[RJ96] Lars Rasmusson and Sverker Jansson. Simulated social control for secure internet commerce (position paper). In *New Security Paradigms Workshop*, 1996.

[Rot67] Julian B. Rotter. A new scale for the measurement of interpersonal trust. *Journal of Philosophy*, 35:651–665, 1967.

[RS97] Michael Reiter and Stuart Stubblebine. Toward acceptable metrics of authentication. In *IEEE Symposium on Security and Privacy*, 1997.

[RT99] Elizabeth Royer and C-K Toh. A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Communications Magazine*, pages 46–55, April 1999.

[RV97] Paul Resnick and Hal Varian. Recommender systems. *Communications of the ACM*, 40(3), March 1997.

[Sha94] Upendra Shardanand. Social information filtering for music recommendation. Master's thesis, MIT, 1994.

[Sha05] Cosma Rohilla Shalizi. Ergodic theory. http://cscs.umich.edu/ crshalizi/notebooks/ergodic-theory.html, May 2005.

[Shi00] Clay Shirky. In praise of freeloaders. OpenP2P, www.openp2p.com, December 2000.

[Shi03] Clay Shirky. File-sharing goes social. http://www.shirky.com, October 2003.

[SJ93] Nematollaah Shiri and Hasan M. Jamil. Uncertainty as a function of expertise. In *Workshop on Incompleteness and Uncertainty in Information Systems*, 1993.

[SK94] Philippe Smets and Robert Kennes. The transferable belief model. *Artificial Intelligence*, 66:191–234, 1994.

[SK97] Ph. Smets and R. Kruse. The transferable belief model for belief representation. In Motro A. and Smets Ph., editors, *Uncertainty Management in information systems: from needs to solutions*, pages 343–368. Kluwer, Boston, 1997.

[SM95] Upendra Shardanand and Patti Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1, pages 210–217, 1995.

[Sme98] Philippe Smets. The transferable belief model for quantified belief representation. In Gabbay D. M. and Smets Ph., editors, *Handbook of defeasible reasoning and uncertainty management systems*, pages 267–301. Kluwer, Doordrecht, 1998.

[Sni00] Brian T. Sniffen. Trust economies in the Free Haven Project. Master's thesis, MIT, May 2000.

[SvO94] Paul Syverson and Paul van Oorschot. On unifying some cryptographic protocol logics. In *IEEE Computer Society Symposium on Research in Security and Privacy*, 1994.

[Swa92] Julia Swannell, editor. *The Oxford Modern English Dictionary*. Clarendon Press, 1992.

[Tad99] Steve Tadelis. What's in a name? reputation as tradeable assets. *American Economic Review*, 89(3):548–563, June 1999.

[TH92] Anas Tarah and Christian Huitema. Associating metrics to certification paths. In *European Symposium on Research in Computer Science (ESORICS)*, 1992.

[Tho84] Ken Thompson. Reflections on trusting trust. *Communication of the ACM*, 27(8):761–763, August 1984.

[tru] Trust competition testbed. http://www.lips.utexas.edu/ kfullam/competition/.

[TT03] Domenico Talia and Paolo Trunfio. Toward a synergy between p2p and grids. *IEEE Internet Computing*, July/August 2003.

[Tur02] Jonathan Turner. *The Structure of Sociological Theory*. Wadsworth Publishing, 2002.

[Van01] Ashlee Vance. First P2P virus hits. *http://security.itworld.com*, February 2001.

[web] Neurogrid website. www.neurogrid.net/.

[Win] Dave Winer. RSS 2.0 specification. http://blogs.law.harvard.edu/tech/rss.

[WJ95] Michael Wooldridge and Nick Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2), 1995.

[WS98] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, June 1998.

[Yam98] Toshio Yamagishi. *The structure of trust*. Tokyo University Press, 1998.

[YKB93] Raphael Yahalom, Birgit Klein, and Thomas Beth. Trust relationships in secure systems - a distributed authentication perspective. In *IEEE Symposium on Security and Privacy*, 1993.

[YS02] Bin Yu and Munindar P. Singh. Distributed reputation management for electronic commerce. *Computational Intelligence*, 18(4):535–549, 2002.

[YS03] Bin Yu and Munindar P. Singh. Searching social networks. In *Proceedings of Second International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2003.

[Zim94] Phil Zimmermann, editor. *PGP User's Guide*. MIT Press, Cambridge, 1994.